Bundle Methods for Regularized Risk Minimization with Applications to Robust Learning

Choon Hui Teo

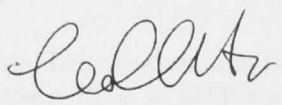
A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

May 2010

Declaration

This thesis is an account of research undertaken between March 2006 and November 2009 at the College of Engineering and Computer Science, the Australian National University, Canberra, Australia.

Except where otherwise indicated, this thesis is my own original work.



Choon Hui Teo

May 2010

Acknowledgements

This thesis would not have been possible without the generous scholarships from NICTA and ANU, and without the help and support of many friends, colleagues, and my family.

Foremost, I would like to thank my ex-supervisory panel chair S.V.N. (Vishy) Vishwanathan. For the past four years, Vishy has been an exemplary teacher and a very supportive collaborator. I believe there are very few, if any, advisors like Vishy who is willing to help debugging his students' code at late nights and to teach his students during weekends. I would also like to thank my advisor, Alex Smola, for sharing his wealth of brilliant ideas and for his guidance. I extend warm thanks to Wray Buntine for his support and help.

I would like to thank Amir Globerson, Sam Roweis, Leon Bottou, and Hans-Peter Graf for being kind and friendly hosts while I was visiting them at MIT and at NEC Labs America. I learned a lot during these visits. I would also like to thank Aleksander Kołcz from Microsoft Research for his guidance while I was interning at Microsoft Live Labs.

I am especially grateful to Yong Haur Tay and Wen Kin Lai. They brought me into the world of AI and machine learning and showed me how interesting and challenging it is to do research in these areas.

Of course, my time as a student would not be as wonderful without my collaborators, my colleagues at NICTA, and the people I met during academic visits. These amazing people I would like to thank here are: Douglas Aberdeen, Marconi Barbosa, Justin Bedo, Antoine Bordes, Karsten Borgwardt, Olivier Buffet, Simon Burton, Tiberio Caetano, Olivier Chapelle, Li Cheng, Ronan Collobert, Chuong (Tom) Do, Gideon Drior, Kenji Fukumizu, Kishor Gawande, Arthur Gretton, Simon Günter, Omri Guttman, Markus Hegland, Marcus Hutter, Dmitry Kamenetsky, Alexandros Karatzoglou, Adam Kowalczyk, Quoc Le, Julian McAuley, Ding Nan, Brian Parker, James Petterson, Novi Quadrianto, Mark Reid, Conrad Sanderson, Scott Sanner, Bernhard Schökolpf, Nic Schraudolph, Tim Sears, Ankan Shaha, Qinfeng (Javen) Shi, Le Song, Peter Sunehag, Owen Thomas, Ivor Tsang, Chris Weber, Markus Weimer, Jin Yu, and Xinhua Zhang.

I would like to thank Michelle Moravec, Debbie Pioch, and Suzzanne Vanhaeften. Their help greatly reduced the amount of time I spent in administrative work so that I had more time to do research.

Finally, I am incredibly grateful to my parents, wife, brothers, and sisters for their love and encouragement which keep me moving forward.

Abstract

Supervised learning in general and regularized risk minimization in particular is about solving optimization problem which is jointly defined by a performance measure and a set of labeled training examples. The outcome of learning, a model, is then used mainly for predicting the labels for unlabeled examples in the testing environment.

In real-world scenarios, a typical learning process often involves solving a sequence of similar problems with different parameters before a final model is identified. For learning to be successful, the final model must be produced timely, and the model should be robust to (mild) irregularities in the testing environment. The purpose of this thesis is to investigate ways to speed up the learning process and improve the robustness of the learned model.

We first develop a batch convex optimization solver specialized to the regularized risk minimization based on standard bundle methods. The solver inherits two main properties of the standard bundle methods. Firstly, it is capable of solving both differentiable and non-differentiable problems, hence its implementation can be reused for different tasks with minimal modification. Secondly, the optimization is easily amenable to parallel and distributed computation settings; this makes the solver highly scalable in the number of training examples.

However, unlike the standard bundle methods, the solver does not have extra parameters which need careful tuning. Furthermore, we prove that the solver has faster convergence rate. In addition to that, the solver is very efficient in computing approximate regularization path and model selection.

We also present a convex risk formulation for incorporating invariances and prior knowledge into the learning problem. This formulation generalizes many existing approaches for robust learning in the setting of insufficient or noisy training examples and covariate shift. Lastly, we extend a non-convex risk formulation for binary classification to structured prediction. Empirical results show that the model obtained with this risk formulation is robust to outliers in the training examples.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

$\mathbb{R}$	The set of all real numbers
$\mathbb{R}_+$	The set of all non-negative real numbers
$\mathbb{R}^d$	d-dimensional Euclidean space
$\Pr(A)$	Probability of event A
$[k]$	The set of integers $\{1, 2, \ldots, k\}$
$[p:q]$	The set of integers $\{p, p+1, \ldots, q\}$
R	Empirical risk functional
Ω	Regularizer
J	Regularized empirical risk *i.e.*, $J := \Omega + R$
$\bar{J}_{\mathbf{u}}$	First order Taylor approximation (*i.e.*, linearization) of J at point $\mathbf{u}$
$\check{J}_t$	Maximum over t linearizations of J
$\mathring{J}_t$	$\check{J}_t$ augmented with a prox-function
$\check{R}_t$	Maximum over t linearizations of R
J_t	$J_t := \Omega + \check{R}_t$
ϕ	Feature mapping
Δ	Label loss function
$\mathcal{X}, \mathcal{Y}, \mathcal{F}$	Spaces of input, output, and compatibility functions, respectively
f	Compatibility function
$\mathbf{u}, \mathbf{v}$	Vectors in $\mathbb{R}^d$
$\mathbf{v}_i$	i-th vector in a sequence or set of vectors
v_j	j-th component of $\mathbf{v}$
$v_{i,j}$	j-th component of $\mathbf{v}_i$
$\langle \mathbf{u}, \mathbf{v} \rangle$, $\mathbf{u}^\top \mathbf{v}$	Dot product of two vectors $\mathbf{u}$ and $\mathbf{v}$
$\|\mathbf{v}\|_p$	L_p norm, *i.e.*, $\left(\sum_{i=1}^d \|v_i\|^p\right)^{\frac{1}{p}}$
$\mathbf{u} \leq \mathbf{v}$	Means $u_i \leq v_i$ for all $i \in [d]$
$\mathbf{0}_d, \mathbf{1}_d$	d-dimensional vectors with all component being zero and one, respectively
$\mathbf{W}, \mathbf{Q}, \mathbf{G}, \mathbf{A}$	Matrices
$\mathbf{W}_i$	i-th row of matrix $\mathbf{W}$
$\mathbf{W}^j$	j-th column of matrix $\mathbf{W}$
$\mathbf{W}_i^j$	Entry of matrix $\mathbf{W}$ at i-th row and j-th column
$\mathbf{I}(A)$	Indicator function which returns 1 if event A is true and 0 otherwise
$\mathbf{i}_U(\mathbf{w})$	Indicator function which returns 0 if $\mathbf{w} \in U$ and $+\infty$ otherwise
$\partial_{\mathbf{w}} J(\mathbf{u})$, $\partial J(\mathbf{u})$	Subdifferential of a non-differentiable function $J(\mathbf{w})$ at $\mathbf{u}$ *i.e.*, $\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w})\|_{\mathbf{w}=\mathbf{u}}$
$\nabla_{\mathbf{w}} J(\mathbf{u})$, $\nabla J(\mathbf{u})$	Gradient of a differentiable function $J(\mathbf{w})$ evaluated at $\mathbf{u}$
$\nabla^2_{\mathbf{w}} J(\mathbf{u})$ $\nabla^2 J(\mathbf{u})$	Second order derivative of a differentiable function $J(\mathbf{w})$ evaluated at $\mathbf{u}$

Chapter 1

Introduction

Supervised machine learning is a statistical discipline devoted to the study of data structures and algorithms that allow machines to *learn* from examples. Unlike AI, its main goal is not to mimic human intelligence in general, but to extract *usable* knowledge in a concise and comprehensible form from examples *i.e.*, a finite set input-output pairs.

In this thesis, we focus on a learning framework in which the knowledge learned is a compatibility function f which takes an input x and an output y as arguments and gives a score of how *compatible* y is to x. The prediction for an unseen input is then defined as the output that maximizes the compatibility score. The framework we adopt here cast the learning problems into rather well-studied numerical optimization problems.

In this thesis, we develop a general, scalable, and efficient optimization method for the learning problems. The method is general as it is capable of solving many different learning problems corresponding to either differentiable or non-differentiable optimization problems. It is scalable because the algorithm is easily amenable to parallel and distributed computation settings. Moreover, it is efficient in solving a sequence of learning problems with slight changes in the objective functions.

We also present efficient methods for improving the performance of the compatibility function when training data is insufficient or of poor quality (*e.g.*, missing or uncertain features, noise, outliers, etc.) and when there is an unknown adversary in the testing environment.

In the remaining of this chapter, we give some motivating examples of machine learning applications and an outline of the thesis.

1.1 Examples of Supervised Machine Learning Applications

In this section, we give some brief but concrete examples of machine learning applications to clarify the use of compatibility function:

- **Document classification**
  The task here is to classify documents (*i.e.*, a sequence of words) into different categories according to their content. This application is used in many areas ranging from email spam filtering to automatic categorization of online news articles. For example, given a document x, the set of categories $\{1, \ldots, k\}$, and a compatibility function in the form $f(x, y) = \mathbf{w}^\top \phi(x, y)$, where $\mathbf{w} \in \mathbb{R}^d$ is a learned weight vector associated to f and $\phi(x, y) \in \mathbb{R}^d$ is a feature mapping. The predicted category for x is taken to be the integer y^* such that $f(x, y^*) \geq f(x, y)$ for all $y \in \{1, \ldots, k\}$.

- **Document ranking**
  One of the keys to the success of information retrieval system is the ability to rank retrieved results according to their relevance to the retrieval criterion set by the users. An important application area is Internet search engines. In this case, the task is to assign a ranking, *i.e.*, a total ordering to the set of webpages $x := \{x_1, \ldots, x_k\}$ returned by the retrieval subsystem after a user issued a query q. The compatibility function can be of the form $f(x, y) = \sum_{i=1}^{k} y_i \, \mathbf{w}^\top \phi(x_i, q)$, where $\mathbf{w} \in \mathbb{R}^d$ is a learned weight vector associated to f, y is a permutation of the set $\{1, \ldots, k\}$, and $\phi(x_i, q) \in \mathbb{R}^d$ is a feature mapping. It can be shown that the ranking for the set of webpages is a y^* such that $\mathbf{w}^\top \phi(x_{y_1^*}, q), \ldots, \mathbf{w}^\top \phi(x_{y_k^*}, q)$ is in non-increasing order and y can be obtained by sorting the values $\mathbf{w}^\top \phi(x_i, q)$.

- **Sequence labeling**
  The goal in this type of problems is to assign labels to a sequence of objects. For example, in named entity recognition, the names of people, places, organizations, etc. that appear in a piece of text (*i.e.*, a sequence of words) are to be marked so that subsequent processing such as document understanding (*e.g.*, answering questions of the type "who did what where and when") can be performed. Say we are only interested in tagging the name of person as '1' and non-name as '0' for a passage x of k words $(x_1, \ldots, x_k)$. The compatibility function can be in the form $f(x, y) = \mathbf{w}^\top \phi(x, y)$ where $\mathbf{w} \in \mathbb{R}^d$ is a learned weight vector associated to f, $y \in \{0, 1\}^k$ is a possible labeling, and $\phi(x, y) \in \mathbb{R}^d$ is a feature mapping. Under some reasonable modeling assumptions such as the label of i-th word only depends on $(i-1)$-th and $(i+1)$-th words, the predicted labeling

$$y^* := \operatorname*{argmax}_{y' \in \{0,1\}^k} \mathbf{w}^\top \phi(x, y').$$

can be computed efficiently via dynamic programming.

The feature mapping ϕ is a device that human experts design to capture the features of a particular problem domain. For instance, in document classification, ϕ is simply a vector of frequency counts of all unique words found in a document.[1]

In this thesis, we focus on a learning framework, namely regularized risk minimization (RRM) that cast learning problems as numerical optimization. While in-depth discussion on RRM is given in Section 2.1, we briefly state the optimization problem here for reference : Under the RRM framework, the compatibility function f is obtained via the following:

$$f = \operatorname*{argmin}_{f \in \mathcal{F}} \lambda\Omega(f) + R(f) \quad \text{where} \quad R(f) := \sum_{i=1}^{m} l(x_i, y_i, f),$$

$\mathcal{F}$ is the space of compatibility function, Ω is a regularizer which penalizes overly complex f, $\lambda > 0$ is a regularization parameter which controls the magnitude of the penalization of f, $\{(x_i, y_i)\}_{i=1}^{m}$ are training input-output pairs, and $l(x, y, f)$ is a loss function which measures the discrepancy between the actual output y_i and predicted output y_i^* for input x_i.

There are two main aims to achieve in this thesis: Firstly, we want to develop a method specialized to RRM problems which is general, easy to implement, scalable, and has faster convergence rate than standard methods. The second aim is to design a general formulation of loss functions which allows incorporation of invariances and prior knowledge for improving the performance of compatibility in the testing environments.

1.2 Thesis Outline

Below is the summary of the chapters in this thesis:

Chapter 2. Background: In this chapter, we review the regularized risk minimization framework for discriminative learning and briefly describe many state of the art machine learning algorithms for solving problems cast into the regularized risk minimization framework. We also review a family of bundle methods for minimizing non-differentiable convex functions. The bundle methods will serve as the basis of our method developed in Chapter 3.

Chapter 3. Bundle Method for Regularized Risk Minimization (BMRM): In this chapter we focus on convex regularized risk minimization. In this case, the

[1] Throughout this thesis we assume the problem dependent ϕ is given if not specified explicitly.

objective function is a sum of two convex functions, namely, regularizer and empirical risk. Instead of building an approximation for the whole objective as in standard bundle methods, our method approximates only the empirical risk. This decoupling allows us to obtain a faster convergence rate. Despite the modifications, our method still inherits many properties of the standard bundle methods which are crucial for an efficient, modular, and scalable (*i.e.*, under parallel and distributed computation model) implementation. We also present experiments to evaluate the performance of our methods under different training settings and against other methods.

Chapter 4. Efficient Computation of Regularization Path: In a typical machine learning process, multiple regularization parameters are tried for two purposes: Firstly, regularization parameter fully identify a learned model for a fixed training data and learning algorithm. Therefore, model selection is done by finding the regularization parameter for which the model achieves the best performance on a hold-out dataset. Secondly, the learned model, *e.g.*, a vector of feature weights, corresponding to different regularization parameters can be used for studying the evolution of the feature weights subject to different regularization magnitudes. The method developed in Chapter 3 builds an approximation of the empirical risk which is independent of the regularizer and the regularization parameter. We present experiments to show that by reusing the approximation of empirical risk corresponding to a regularization parameter, the learning with a new regularization parameter can be sped up significantly.

Chapter 5. Robust Learning with Invariances: In this chapter, we present a worst-case convex learning formulation for incorporating invariances and prior knowledge about a problem domain into the learning algorithms in a principled way. This formulation generalizes many existing heuristic and principled approaches under a unified setting. We also develop a closely related formulation for dealing with an adversary in the testing environment. We present experiments to show that the resulting model is more robust compared to that obtained by standard learning formulations.

Chapter 6. Non-convex Loss for Structured Prediction: In this chapter, we present a modification of a non-convex empirical risk, originally developed for binary classification, to structured prediction problems. We show experimentally that the resulting formulation is robust to noisy data.

Chapter 7. Conclusion: In this chapter, we summarize our contributions and provide a discussion on potential future work.

Appendix A. Loss Functions: This appendix provides a discussion on many convex loss functions commonly used in machine learning applications. The presenta-

tion/formulation of the loss functions is suited to the regularized risk minimization framework.

Appendix B. Proofs: This appendix keeps long proofs developed in this thesis.

Appendix C. Basic Definitions and Results in Convex Analysis: This appendix collects some basic definitions and results in convex analysis that we make use of throughout the thesis.

Chapter 2

Background

In this chapter, we review a principled learning framework, namely regularized risk minimization (RRM) [Vapnik, 1995] which justifies how and why the learning of a compatibility function f can be done via standard numerical optimization. Then we describe a number of state-of-the-art machine learning algorithms for solving problems cast as RRM. Finally, we review a class of non-differentiable optimization methods, namely, bundle methods [see *e.g.*, Hiriart-Urruty and Lemaréchal, 1993] which are suitable for solving various types of RRM.

2.1 Machine Learning as Risk Minimization

In this section, we see how a machine learning problem can be cast in the form of (risk) minimization problem. Formally, we consider an input space $\mathcal{X}$, an output space $\mathcal{Y}$ and a space $\mathcal{F}$ of compatibility functions. In the supervised learning setting, we are given a set of *training* examples, *i.e.*, a set of input-output pairs $S := \{(x_1, y_1), \ldots, (x_m, y_m)\} \subset \mathcal{X} \times \mathcal{Y}$. Furthermore, we assume that the examples in S are independently and identically distributed (iid) according to an unknown distribution P on the space $\mathcal{X} \times \mathcal{Y}$.

We define three functions that are key to the formulation of risk minimization as follows:

- Compatibility function

  $$f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

  measures the compatibility of an input-output pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, *i.e.*, the higher the value $f(x, y)$ the more compatible the pair (x, y) is.

- Prediction function

  $$y^* : \mathcal{X} \times \mathcal{F} \to \mathcal{Y}$$

predicts an output $y \in \mathcal{Y}$ for a given input $x \in \mathcal{X}$ under the guidance of a compatibility function $f \in \mathcal{F}$. For instance, y^* can be defined as maximization of compatibility function over the output space, for a fixed input, *i.e.*,

$$y^*(x, f) := \underset{y \in \mathcal{Y}}{\operatorname{argmax}} f(x, y).$$

- Loss function

$$l : \mathcal{X} \times \mathcal{Y} \times \mathcal{F} \to \mathbb{R}_+$$

measures the level of disagreement between target and predicted outputs. For instance, in classification, l can be defined as the so-called 0/1 loss:

$$l(x, y, f) = \begin{cases} 0 & : \quad y = y^*(x, f) \\ 1 & : \quad \text{otherwise} \end{cases}, \tag{2.1}$$

and in regression (*i.e.*, $\mathcal{Y} = \mathbb{R}$), l can be defined as the squared absolute difference of target and predicted outputs:

$$l(x, y, f) = |y - y^*(x, f)|^2.$$

In fact, the loss function also measures the quality of a compatibility function as the prediction is solely depending on the latter. In other words, the loss incurred by a "better" compatibility function f_1 is, on average, lower than that by a "worse" one f_2. More formally, it means that the risk of f_1 is lower than the risk of f_2, where the risk of f is defined as the expected loss of f over the distribution P:

$$Risk(f) := \int_{\mathcal{X} \times \mathcal{Y}} l(x, y, f) P(x, y) \, dx \, dy. \tag{2.2}$$

It follows that learning the optimum compatibility function $f_{\mathcal{F}} \in \mathcal{F}$ is equivalent to solving the following optimization problem:

$$f_{\mathcal{F}} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, Risk(f). \tag{2.3}$$

Since the distribution P is unknown by assumption, (2.2) cannot be evaluated and hence problem (2.3) is not solvable. Nevertheless, in Sections 2.1.1, 2.1.2, and 2.1.3, we describe how the risk (2.2) can be estimated and minimized with theoretical guarantees on the quality of estimation.

2.1.1 Empirical Risk Minimization (ERM)

Since we are given an iid sample S, the risk (2.2) can be estimated by the *empirical risk* – an empirical average computed on S:

$$R(f) = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, f).$$

Suppose $l(x, y, f)$ is bounded for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and all $f \in \mathcal{F}$ (which is often the case *e.g.*, the 0/1 loss (2.1)), then with the iid assumption on S, $l(x, y, f)$ can be regarded as iid random variables too, for any f. Therefore, one can show that, for any f

$$R(f) \to Risk(f) \quad \text{as} \quad m \to \infty, \tag{2.4}$$

by the law of large numbers. The convergence (2.4) explains only why $Risk(f)$ can be estimated by $R(f)$ when f is fixed; it does not answer the question on how $R(f)$ can be used to obtain a compatibility function. If one finds a compatibility function f_m on S via

$$f_m = \underset{f \in \mathcal{F}}{\operatorname{argmin}} R(f), \tag{2.5}$$

then it is possible that f_m *memorizes* all the examples in S but is not able to provide sensible scores for examples not seen in S. For instance, $y^*(x_i, f_m) = y_i \; \forall (x_i, y_i) \in S$ and, for some fixed $y_0 \in \mathcal{Y}$, $y^*(x, f_m) = y_0 \; \forall (x, y) \in \mathcal{X} \times \mathcal{Y} \setminus S$. This phenomenon is generally known as *overfitting*.

A more formally defined statistical notion that includes overfitting as a special case is *consistency* [Vapnik, 1995]. The consistency of ERM with respect to $\mathcal{F}$ is defined as: for all $\epsilon > 0$

$$\Pr\left(Risk(f_m) - Risk(f_{\mathcal{F}}) > \epsilon\right) \to 0 \quad \text{as} \quad m \to \infty.$$

Vapnik and Chervonenkis [1971, 1991] showed that uniform convergence *i.e.*,

$$\Pr\left(\sup_{f \in \mathcal{F}} |Risk(f) - R(f)| > \epsilon\right) \to 0 \quad \text{as} \quad m \to \infty \tag{2.6}$$

is a necessary and sufficient condition for ERM to achieve consistency. Intuitively, condition (2.6) implies that the function space $\mathcal{F}$ must not contain functions which just "memorize" training examples for all $m > 0$. Otherwise, the difference between the true and empirical risks may not get arbitrary small for all functions $f \in \mathcal{F}$.

Indeed, this is clarified through the generalization bound (*i.e.*, upper bound on the

true risk) [Bousquet et al., 2004, Vapnik, 1995]: for any function $f \in \mathcal{F}$,

$$Risk(f) \leq R(f) + 2\sqrt{2\frac{h \log \frac{2em}{h} + \log \frac{2}{\delta}}{m}} \tag{2.7}$$

with probability at least $1 - \delta$, where h is the Vapnik-Chervonenkis (VC) dimension[1] which indicates the capacity of $\mathcal{F}$, m is the size of sample S, and e is the exponential constant. From (2.7) we see that the function space $\mathcal{F}$ which prevents overfitting must have finite VC dimension $h < \infty$ [von Luxburg and Schölkopf, 2009] so that the second term of (2.7) will approach 0 and $R(f)$ will approach $Risk(f)$ as sample size m grows.

We note that the bound (2.7) is rather loose as it considers only the worst function $f \in \mathcal{F}$ and is independent of the distribution P. Tighter bounds can be obtained by using more advanced analysis techniques which require further assumptions or knowledge on the distribution P or data S (see *e.g.*, [Mendelson, 2003, Bousquet et al., 2004, von Luxburg and Schölkopf, 2009] and references therein).

2.1.2 Structural Risk Minimization (SRM)

In ERM, the function space $\mathcal{F}$ is required to be fixed *a priori* and the ratio of sample size over VC dimension of $\mathcal{F}$, m/h, be large so that the capacity term *i.e.*, second term on the right hand side of (2.7), becomes small. Hence, small value of empirical risk guarantees a small value of the risk, that is, the minimizer f_m of empirical risk (2.5) has good generalization ability.

In practice, the ratio m/h may be small due to small sample given or large function space chosen. In this case, small empirical risk does not guarantee a small risk. To minimize the risk (indirectly), one would have to keep the capacity of $\mathcal{F}$ as small as possible while minimizing the empirical risk.

SRM [Vapnik, 1995] achieves this by augmenting the empirical risk with a penalty term $\text{pen}(\mathcal{F}, m)$ that is proportional to the capacity (*e.g.*, VC dimension) of $\mathcal{F}$ and inversely proportional to the sample size m. Then, it considers an infinite sequence of nested function spaces: $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots$ and learns the best $f_m \in \cup_{i \in \mathbb{N}} \mathcal{F}_i$ via the following minimization problem:

$$f_m = \underset{f \in \mathcal{F}_n,\ n \in \mathbb{N}}{\operatorname{argmin}} \ R(f) + \text{pen}(\mathcal{F}_n, m). \tag{2.8}$$

[1] The VC dimension of a function space $\mathcal{F}$ is defined as the largest number h such that there exists a sample S of size h for which functions in $\mathcal{F}$ can realize any possible labeling to it [Vapnik, 1995].

2.1.3 Regularized Risk Minimization (RRM)

A popular alternative to (2.8) is a regularization method that controls the complexity of a function $f \in \mathcal{F}$ instead of the capacity of the function space $\mathcal{F}$. The complexity of f is measured by a non-negative lower semi-continuous function $\Omega : \mathcal{F} \to \mathbb{R}_+$ [Vapnik, 1995, Hiriart-Urruty and Lemaréchal, 1993]. The resulting "constrained" formulation

$$f_m^\tau = \operatorname*{argmin}_{f \in \mathcal{F}} R(f) \quad \text{subject to} \quad \Omega(f) \leq \tau \tag{2.9}$$

with $\tau > 0$ shares the same spirit of SRM when carried out over a sequence of bounds τ_i on $\Omega(f)$ in an increasing order: $0 < \tau_1 < \tau_2 < \ldots$. That is, (2.9) is performed over the sequence of spaces: $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots$, with $\mathcal{F}_k := \{f : \Omega(f) \leq \tau_k\}$. Unlike (2.8), the best compatibility function f_m^τ is determined by its prediction performance on a set of examples not seen in training set S (see cross validation [Duda et al., 2001]).

An "unconstrained" formulation of (2.9) that resembles Tikhonov's regularization method [Vapnik, 1995, Tikhonov and Arsenin, 1977] reads:

$$f_m^\lambda = \operatorname*{argmin}_{f \in \mathcal{F}} \lambda\Omega(f) + R(f) \tag{2.10}$$

with $\lambda > 0$. In (2.9), $R(f_m^\tau)$ decreases as τ is increased (and the feasible function space $\mathcal{F}$ expands), whereas in (2.10), $R(f_m^\lambda)$ decreases as λ is decreased because the term $\lambda\Omega(f)$ is outweighed by the term $R(f)$ in the minimization. Therefore, λ in (2.10) is inversely proportional to τ in (2.9). We also note that it is common to write (2.10) in the form:

$$f_m^C = \operatorname*{argmin}_{f \in \mathcal{F}} \Omega(f) + CR(f)$$

with $C = \lambda^{-1} > 0$. Moreover, $f_m^C = f_m^\lambda$ as positive scaling changes only the shape of objective function but not the location of its minimizer.

2.1.4 Convex Regularized Risk

We have seen how learning is reduced to numerical optimization of regularized risk. In general, the optimization task may not be easy to carry out because the regularized risk could be highly nonlinear, or, worse still, discontinuous. To avoid such computational difficulties, the regularized risk is usually restricted to the set of convex functions (see Definition C.0.6) which can be solved efficiently with global optimality guarantee [Boyd and Vandenberghe, 2004].

Throughout this thesis, we refer to the function Ω and the constant λ introduced in Section 2.1.3 as regularizer and regularization constant, respectively. Furthermore, we

restrict the compatibility function space $\mathcal{F}$ to the set of functions parameterized by a d-dimensional weight vector $\mathbf{w} \in \mathbb{R}^d$ with d a positive integer. The input and output spaces $\mathcal{X}, \mathcal{Y}$ remain arbitrary as long as a feature mapping $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is provided. In particular, the compatibility function is defined as a weighted sum of features (*cf.* the examples in the beginning of this chapter):

$$f(x, y) = \mathbf{w}^\top \phi(x, y)$$

for some input-output pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and some feature mapping ϕ. Since each $f \in \mathcal{F}$ is uniquely identified by a corresponding weight vector $\mathbf{w} \in \mathbb{R}^d$, we rewrite the regularizer $\Omega(f)$, the empirical risk $R(f)$, and the loss function $l(x, y, f)$ as $\Omega(\mathbf{w})$, $R(\mathbf{w})$, and $l(x, y, \mathbf{w})$, respectively, whenever the context is clear.

Note that restricting regularized risk to a convex function and parameterizing the function space with finite dimensional weight vector are not unreasonable assumptions in machine learning. In fact, many state-of-the-art learning algorithms are instances of this setting. Moreover, these algorithms have been successfully applied to a variety of problem domains. The examples include support vector machine (SVMs) [Boser et al., 1992, Cortes and Vapnik, 1995, Schölkopf and Smola, 2002] for binary classification, support vector regression (SVR) [Vapnik et al., 1997] for univariate regression, support vector description [Tax and Duin, 1999] or novelty detection [Schölkopf et al., 2001] for detecting outliers in a database, conditional random fields (CRFs) [Lafferty et al., 2001] for sequence labeling, and max-margin Markov networks (M^3Ns) [Taskar et al., 2004] for handwriting recognition, to name a few.

Convex Regularizer

There are many convex functions suitable for use as regularizers. One of the most commonly used regularizers is the squared L_2 norm:

$$\Omega(\mathbf{w}) := \|\mathbf{w}\|_2^2 = \mathbf{w}^\top \mathbf{w} \,. \tag{2.11}$$

This regularizer has gained popularity since the introduction of max-margin methods such as SVMs [Boser et al., 1992]. In these methods, the norm $\|\mathbf{w}\|_2$ is inversely proportional to the "margin" that is maximized to achieve generalization. Furthermore, this regularizer turns many of those methods into well-studied quadratic programming problems [Boyd and Vandenberghe, 2004].

The regularizer (2.11) can be generalized by a quadratic form regularizer *i.e.*,

$$\Omega(\mathbf{w}) := \mathbf{w}^\top \mathbf{M}\, \mathbf{w} \tag{2.12}$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a positive definite matrix *i.e.*, $\mathbf{u}^\top \mathbf{M}\, \mathbf{u} > 0,\ \forall\, \mathbf{u} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$. We see that (2.11) is a special case of (2.12) where $\mathbf{M}$ is equal to the identity matrix.

The quadratic form regularizer is particularly useful when the components of $\mathbf{w}$ are to be regularized at different magnitudes (instead of uniform magnitude in (2.11)). Sandler et al. [2009] showed that this type of regularizer is useful for incorporating prior information.

The L_1 norm regularizer

$$\Omega(\mathbf{w}) := \|\mathbf{w}\|_1$$

is also commonly used in high dimensional problems due to its sparsity-inducing property [Tibshirani, 1996]. That is, the weights for irrelevant features are automatically shrunk to zero and hence the weight vector $\mathbf{w}$ will contain few non-zero components. The resulting weight vector with few non-zero components eases the interpretation of the compatibility functions or the importance of the features.

Convex Risk

The risk (2.2) is defined as an expected loss of compatibility function f over examples (x, y) drawn from P. The loss l may be an exact measure of prediction performances such as misclassification error (*i.e.*, 0/1 loss), area under ROC curve, F-measure, etc. [Joachims, 2005] which are discontinuous over the compatibility function space $\mathcal{F}$. In general, the discontinuity of the loss function renders the risk minimization computationally intractable.

To remedy this difficulty, most prominent machine learning methods such as the SVM and its variants [see *e.g.*, Bennett and Mangasarian, 1992, Joachims, 2005, Tsochantaridis et al., 2005] use non-negative convex (and hence continuous) surrogate loss functions which upper bound the original discontinuous loss functions.[2] The convexity of the loss function comes into the picture as it provides many attractive properties such as guarantee of global optimality and availability of efficient algorithms [Hiriart-Urruty and Lemaréchal, 1993, Boyd and Vandenberghe, 2004] for the risk minimization. We briefly describe here two common types of surrogate loss functions used in classification/structured prediction tasks [Taskar et al., 2004, Tsochantaridis et al., 2005, Bakir et al., 2007]: logistic loss

$$l(x, y, f) = \log\left(\sum\nolimits_{y' \in \mathcal{Y}} \exp(f(x, y'))\right) - f(x, y), \tag{2.13}$$

[2] For classification, [Bartlett et al., 2006] has established the quantitative relationship between 0/1 risk (*i.e.*, risk as assessed with 0/1 loss) and surrogate risk (*i.e.*, risk as assessed with surrogate loss) and conditions under which minimization of surrogate risk would lead to the same result as with 0/1 risk.

and max-margin loss

$$l(x, y, f) = \max_{y' \in \mathcal{Y}} \Gamma(y, y')[f(x, y') - f(x, y)] + \Delta(y, y'), \tag{2.14}$$

where $\Gamma(y, y') \geq 0$ is a margin scaling term which specifies how large the margin should be enforced and $\Delta(y, y') \geq 0$ is the cost of misclassifying y by y'. See Appendix A for a more comprehensive discussion on many widely used loss functions.

In the case of binary classification where $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{1, -1\}$, and $f(x, y) := \frac{1}{2} y \mathbf{w}^\top x$, where $\mathbf{w} \in \mathbb{R}^d$, (2.14) is equivalent to the hinge loss in SVMs [Cortes and Vapnik, 1995]:

$$l(x, y, \mathbf{w}) = \max(0, 1 - y \mathbf{w}^\top x). \tag{2.15}$$

and (2.13) is equivalent to the logistic (regression) loss [Collins et al., 2000]:

$$l(x, y, \mathbf{w}) = \log(1 + \exp(-y \mathbf{w}^\top x)). \tag{2.16}$$

Figure 2.1 illustrates the 0/1 loss and its two convex upper bounds: hinge and logistic loss functions. From the figure, we see that the convex upper bounds are continuous in the values of $f(x, y)$, that is, without gap or jumps in the graphs.

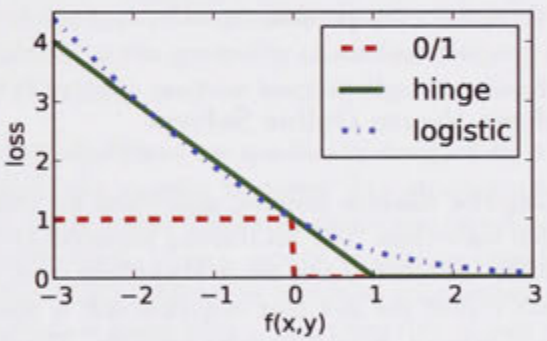


Figure 2.1: 0/1 loss and its convex upper bounds: hinge loss and logistic loss.

Generally, all loss functions can be categorized as differentiable (*e.g.*, logistic loss) and non-differentiable (*e.g.*, max-margin loss) [Nocedal and Wright, 1999] when it comes to the selection of optimization methods for solving the risk minimization problem. This is because many optimization methods such as the Newton's method [Nocedal and Wright, 1999] exploit the smoothness (*e.g.*, second order derivatives) of a loss function to achieve fast convergence rate. For the non-differentiable case, where only subgradients [Rockafellar, 1970] of a function are available, many clever strategies have been proposed; we discuss this further in the following section.

Replacing the discontinuous loss function by a convex surrogate and restricting the regularizer to a convex function, we obtain a convex regularized risk minimization

framework which is fundamentally the formulation used in many successful machine learning methods.

2.2 Existing Solvers for Convex Regularized Risk Minimization

In this section, we focus on the case of finite dimensional weight vector $\mathbf{w}$ albeit some of the solvers we discuss here are capable of handling the infinite dimensional case by using the "kernel trick" [Schölkopf and Smola, 2002].

Let the regularized risk be denoted by $J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w})$ and let the minimizer be $\mathbf{w}^* := \operatorname{argmin}_{\mathbf{w}} J(\mathbf{w})$. Most of the existing machine learning algorithms for solving RRM are iterative in nature, that is, they generate a sequence of iterates (*i.e.*, intermediate weight vectors): $\mathbf{w}_1, \mathbf{w}_2, \ldots$ and terminate with a solution $\mathbf{w}_t$ which is ϵ-accurate:

$$\min_{i \in [t]} J(\mathbf{w}_i) - J(\mathbf{w}^*) \leq \epsilon \tag{2.17}$$

where $[t] := \{1, \ldots, t\}$, and $\epsilon > 0$ is pre-defined.

2.2.1 Batch Solvers Versus Online Solvers

It is common to categorize machine learning algorithms for solving RRM as either batch or online. Batch algorithms "see" all training examples in each iteration before generating a new iterate. In other words, batch algorithms treat the empirical risk as a function and do not exploit the fact that empirical risk is decomposable *i.e.*, as a weighted sum of loss computed on each training example. The downside of this type of algorithms is that they have at least linear runtime (to convergence) dependency on the training set size [Joachims, 2006]. Nevertheless, certain batch algorithms can ameliorate this limitation by parallel and distributed computation (assuming the number of computing nodes is linear in the number of examples).

On the contrary, online algorithms exploit the decomposability of empirical risk and generate a new iterate after seeing a training example.[3] Shalev-Shwartz et al. [2007] and Bottou and Bousquet [2007] showed that the convergence of online algorithms is independent of the training set size.[4] In fact, Shalev-Schwartz and Srebro [2008] showed

[3] Certain types of online algorithms such as the subgradient methods [Nedić, 2002, Shalev-Shwartz et al., 2007] can be amended to operate in batch setting simply by aggregating the subgradients of loss computed on each training example and using the aggregate subgradient to compute new iterate.

[4] Note that an iteration refers to the time a new iterate is being generated. Therefore, batch algorithms see all m training examples in 1 iteration whereas online algorithms see m training examples in m iteration, or, in other terminology, 1 epoch.

that the runtime can be inversely proportional to the training set size subject to a fixed generalization error (*i.e.*, a fixed true risk value (2.2)). In practice, there are results showing that online algorithms converge faster than batch algorithms in many problem domains [see *e.g.*, Shalev-Shwartz et al., 2007, Bordes et al., 2007, 2008]. Arguably, the limitations of most online algorithms are that they rely heavily on the randomness in the ordering of training examples [Bordes et al., 2007] and that they are not suitable at all for indecomposable empirical risk such as the multivariate performance measures proposed by Joachims [2005].

All in all, there is no absolute measure for determining whether online algorithms are better than batch algorithms or vice versa. This is because the comparison outcome is dependent on many environment variables such as the training set size, type of access to training data, type of regularizer and empirical risk used, computation time limit, computing resources and environment (*e.g.*, desktop machine vs. large scale cluster), etc.. Therefore, a fair comparison must takes all these factors into consideration.

2.2.2 What Type of Regularized Risk Can Existing Solvers Handle?

Instead of focusing on the type of iterate updates (*i.e.*, batch vs. online), we turn our attention to the question on the generality of machine learning algorithms *i.e.*, what types of regularized risks can a machine learning algorithm handle.

Some machine learning algorithms are specifically designed for a restricted set of regularizers and empirical risks in order to exploit their structures and obtain faster convergence rate. While other machine learning algorithms do not require the regularized risk to be in any specific form. In fact, generic algorithms consider regularized risk as a "black box" function which returns the primitive information such as the value and derivative of the function at query point. This generality usually comes with a price of slower convergence.

It is then natural to ask whether we can find a middle ground between the two extremes *i.e.*, an algorithm which achieves faster convergence than generic algorithms by some mild exploitation of the regularized risk but does not lose (much of) the ability to handle a wider set of regularized risks. Chapter 3 is devoted to the development of one such algorithm, namely, BMRM.

Before we proceed to the description of the proposed algorithm, we review some widely used machine learning algorithms grouped by the type of regularizer and empirical risk they focus on. We first label some commonly used regularized risks as combinations of regularizers and empirical risks. (See Table 2.1.) In particular, we focus on max-margin (2.14) and logistic (2.13) types of risk and L_1 norm and squared L_2 norm regularizers. For other types of risk and regularizer, we group them under the "general" category and discuss their corresponding algorithms collectively.

		$R(\mathbf{w})$		
		max-margin	logistic	general
$\Omega(\mathbf{w})$	$\|\mathbf{w}\|_2^2$	J1	J2	J3
	$\|\mathbf{w}\|_1$	J4	J5	J6
	general	J7	J8	J9

Table 2.1: The entries J1 ... J9 are the labels for combinations of different regularizers $\Omega(\mathbf{w})$ and empirical risks $R(\mathbf{w})$. "general" refers to any convex regularizer or empirical risk. "max-margin" and "logistic" refer to empirical risks with loss functions of type (2.14) and (2.13), respectively.

SMO-M^{3}N [Taskar et al., 2004] solves the dual problem of J1 (*i.e.*, a quadratic program (QP)) with a modified sequential minimal optimization (SMO) [Platt, 1999] which optimizes two dual variables at a time analytically.

SVM$^{\text{struct}}$ [Tsochantaridis et al., 2005], in particular, the "1-slack" formulation SVM$^{\text{perf}}$ [Joachims et al., 2009], solves J1 by using a bundle method [Hiriart-Urruty and Lemaréchal, 1993] (also see Section 2.3) which was shown to converge in $O(1/\epsilon)$ iterations [Teo et al., 2007, Smola et al., 2008, Joachims et al., 2009].

EG [Collins et al., 2008] solves J1 [Bartlett et al., 2005] and J2 [Globerson et al., 2007] via their duals using an exponentiated gradient method [Kivinen and Warmuth, 1994] with an assumption that the output $y \in \mathcal{Y}$ can be decomposed into a polynomial number of parts. It has been shown that the algorithm converges in $O(1/\epsilon)$ and $O(\log(1/\epsilon))$ iterations in the cases of J1 and J2, respectively.

Pegasos [Shalev-Shwartz et al., 2007] is a online subgradient method which solves J3 with a convergence rate of $\tilde{O}(1/\epsilon)$.

The stochastic gradient descent [Bordes et al., 2009] for SVM, SGDSVM solves J3. A closely related method, SGD-QN, which maintains a diagonal approximation to the Hessian of objective function deals with L_2 norm regularizer and twice differentiable loss function, hence it can be categorized as solving J2.

LaRank [Bordes et al., 2007] is an online algorithm that solves the dual of J1 using the SMO style optimization to adjust the values of two dual variables at one time. Unlike SMO, LaRank does not keep/compute gradients corresponding to all training examples in the selection of dual variable pair to optimize. Instead, it keeps a minimal amount of gradients and choose the dual variables pair by alternating between seen and unseen examples.

OCAS [Franc and Sonnenburg, 2008] solves binary classification tasks of type J1 using the "1-slack" SVM$^{\text{struct}}$ with line search step to stabilize the iterates. This modification ensures that the regularized risk is non-increasing as iteration progresses. Although

there is no improvement on the convergence rate, faster runtime has been shown empirically.

LIBLINEAR [Fan et al., 2008] solves binary classification tasks via the dual of J1 using a coordinate descent method [Hsieh et al., 2008] that optimizes one dual variable at a time. This method has been shown [Hsieh et al., 2008] to converge in $O(\log(1/\epsilon))$ iterations. In fact, this method is closely related to Hildreth's method for quadratic programming [Hildreth, 1957, Iusem and Pierro, 1990] and the Passive-Aggressive method [Crammer et al., 2003].

A number of traditional numerical optimization methods [Nocedal and Wright, 1999] have been improved and specialized to solve J5 for high dimensional problems. For example, the interior point method of Koh et al. [2007], the quasi-Newton method of Andrew and Gao [2007], and the coordinate gradient descent method of Tseng and Yun [2009].

Standard linear programming (LP) [see *e.g.*, Vanderbei, 2008] can be used to solve J4. [Zhu et al., 2009] showed a modification of the adaptive ridge regression of Grandvalet [1998] for this type of regularized risk. For high dimensional problem, LP may be inefficient; Duchi et al. [2008] proposed a projected subgradient method for this case with an efficient projection algorithm.

Standard unconstrained numerical optimization techniques for smooth objective function such as the quasi-Newton methods [Nocedal and Wright, 1999] can be used to solve J2.

Subgradient methods [Shor, 1985] assume only the availability of one subgradient of a possibly non-differentiable regularized risk $J(\mathbf{w})$ at a given point in order to compute a new iterate

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \, \mathbf{g}_t$$

where $\mathbf{g}_t \in \frac{\partial}{\partial \mathbf{w}} J(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_t}$, and η_t is a step size which determines the convergence (rate) [Nedić, 2002]. Typically, the convergence rate of subgradient methods is of $O(1/\epsilon^2)$ [Bertsekas, 1995] but for cases such as the Pegasos of Shalev-Shwartz et al. [2007] specialized to J3, faster convergence is achievable. The constrained version of subgradient methods *i.e.*, projected subgradient methods [Bertsekas, 1976] for regularized risk compute a new iterate as follows:

$$\mathbf{w}_{t+1} = \Pi_\Omega(\mathbf{w}_t - \eta_t \, \mathbf{a}_t)$$

where η_t, is a step size, $\mathbf{a}_t \in \frac{\partial}{\partial \mathbf{w}} R(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_t}$, and Π_Ω is a Euclidean projection defined as:

$$\Pi_\Omega(\mathbf{u}) := \operatorname*{argmin}_{\mathbf{w}} \{\|\mathbf{w} - \mathbf{u}\| \;:\; \Omega(\mathbf{w}) \leq \tau\} \tag{2.18}$$

with $\tau > 0$. Duchi and Singer [2009] proposed a forward looking subgradient method which can be regarded as a modification that changes the value of τ as iteration progresses. Recent works [see *e.g.*, Liu et al., 2009, Quattoni et al., 2009] have shown the usefulness of projected subgradient method for J9 with non-differentiable mixed-norm[5] regularizer if the projection step (2.18) can be computed efficiently.

Similar to (projected) subgradient methods, cutting-plane methods [Kelly, 1960, Cheney and Goldstein, 1959] and bundle methods [see *e.g.*, Hiriart-Urruty and Lemaréchal, 1993] require only function value and a subgradient of the regularized risk at a given point. Therefore, cutting-plane and bundle methods can be used to solve J9 and usually converge at a faster (empirical) rate. This is because these methods keep a "bundle" of previous subgradients and exploit as much information as possible from this bundle. We now discuss these methods in more detail.

2.3 Bundle Methods

We have seen in Section 2.1.4 that regularized risks can be either differentiable or non-differentiable depending on the modeling choice (*i.e.*, loss function) made by the users. Specialized algorithms often solve the differentiable or non-differentiable optimization problem faster. However, during the problem modeling phase, it is often that many different loss functions (including both differentiable and non-differentiable) are tried to determine the best loss function to use. In this case, a general algorithm which can handle both differentiable and non-differentiable optimization problems may be more desirable, as long as its time complexity is still acceptable.

In this section, we review bundle methods which were introduced to solve (convex) optimization problems which are not necessarily differentiable. Bundle methods are similar to subgradient methods as they assume only that, when given a point, the value and a subgradient of the function can be evaluated. However, unlike subgradient methods, bundle methods retain previous linearizations (*i.e.*, first order Taylor approximations) and exploit this bundle of linearizations to achieve better convergence. Furthermore, bundle methods provide a natural termination criterion (*i.e.*, upper bound on the gap $J(\mathbf{w}_t) - J(\mathbf{w}^*)$ at iteration t) which does not exist in subgradient methods.

2.3.1 Preliminaries

Throughout this section, we assume the objective function (*i.e.*, regularized risk) $J : \mathbb{R}^d \to \mathbb{R}$ is convex and lower semi-continuous. We note that the convexity of J implies

[5] Examples of popular mixed-norm regularizer include the $L_{1,2}$ norm, $\sum_{i=1}^{d} \left\| \mathbf{W}^i \right\|_2$ and $L_{1,\infty}$ norm, $\sum_{i=1}^{d} \max_j |\mathbf{W}_i^j|$, where $\mathbf{W} \in \mathbb{R}^{d \times k}$, $\mathbf{W}^i \in \mathbb{R}^k$ is the i-th row of $\mathbf{W}$, and $\mathbf{W}_i^j$ is the entry of $\mathbf{W}$ at the intersection of i-th row and j-th column.

that it is locally Lipschitz continuous [Hiriart-Urruty and Lemaréchal, 1993, Theorem IV.3.1.2].

Formally, the subdifferential $\partial J(\mathbf{u})$ of J at point $\mathbf{u}$ is defined to be the set

$$\partial J(\mathbf{u}) := \left\{ \mathbf{g} \ : \ J(\mathbf{w}) \geq J(\mathbf{u}) + \langle \mathbf{w} - \mathbf{u}, \mathbf{g} \rangle \ \ \forall \mathbf{w} \in \mathbb{R}^d \right\}. \tag{2.19}$$

If the set $\partial J(\mathbf{u})$ is not empty then J is said to be *subdifferentiable* at $\mathbf{u}$ and $\mathbf{g} \in \partial J(\mathbf{u})$ a *subgradient* of J at $\mathbf{u}$. (See Figure 2.2 for geometric intuition of subgradient.) On the other hand, if this set is a singleton then the function is said to be *differentiable* at $\mathbf{u}$ and $\mathbf{g} \in \partial J(\mathbf{u})$ the *gradient* of J at $\mathbf{u}$. Convex functions are subdifferentiable everywhere in their domain (see Definition C.0.4) [Hiriart-Urruty and Lemaréchal, 1993].

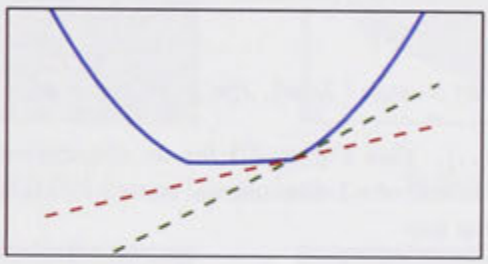


Figure 2.2: Geometric intuition of a subgradient. The non-differentiable 1-dimensional convex function (solid blue) is only subdifferentiable at the "kink" points. Two of its subgradients (dashed green and red lines) at a "kink" point which are tangential to the function. The normal vectors to these lines are subgradients.

2.3.2 Cutting-Plane Method

As implied by (2.19), J is bounded from below by its linearization at any $\mathbf{u} \in \mathbb{R}^d$

$$\bar{J}_{\mathbf{u}}(\mathbf{w}) := J(\mathbf{u}) + \langle \mathbf{w} - \mathbf{u}, \mathbf{g} \rangle$$

for any $\mathbf{g} \in \partial J(\mathbf{u})$ and for all $\mathbf{w} \in \mathbb{R}^d$. More generally, given subgradients $\mathbf{g}_1, \ldots, \mathbf{g}_t$ of J evaluated at points $\mathbf{w}_1, \ldots, \mathbf{w}_t$, the piecewise linear lower bound of J can be stated as the maximum over the linearizations $\bar{J}_{\mathbf{w}_i}$, $i = 1, \ldots, t$:

$$\check{J}_t(\mathbf{w}) := \max_{i \in [t]} \left\{ J(\mathbf{w}_i) + \langle \mathbf{w} - \mathbf{w}_i, \mathbf{g}_i \rangle \right\}.$$

$\check{J}_t$ is convex as it is defined as a pointwise maximum of affine functions [Boyd and Vandenberghe, 2004]. Furthermore, we see that $\check{J}_t$ approximates J better as the number of linearizations increases and the approximation is exact at the points $\mathbf{w}_i$, $i = 1, \ldots, t$.

Cheney and Goldstein [1959] and Kelly [1960] independently developed the cutting-plane method which minimizes the piecewise linear lower bound $\check{J}_t$ instead of J. The key to their algorithms for building $\check{J}_t$ is the following iterate update rule:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in U} \check{J}_t(\mathbf{w}),$$

which can be expressed as a standard linear programming (LP) problem:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\xi \in \mathbb{R},\ \mathbf{w} \in U} \quad \xi \tag{2.20a}$$

$$\text{s.t.} \quad \mathbf{g}_i^\top \mathbf{w} - \xi \leq b_i,\ i = 1, \ldots, t \tag{2.20b}$$

where $U \subset \mathbb{R}^d$ is a convex compact set and $b_i := J(\mathbf{w}_i) - \mathbf{w}_i^\top \mathbf{g}_i$ is an offset. Once a new iterate $\mathbf{w}_{t+1}$ is obtained, the piecewise linear lower bound is updated accordingly, *i.e.*,

$$\check{J}_{t+1}(\mathbf{w}) := \max\left\{\check{J}_t(\mathbf{w}), J(\mathbf{w}_{t+1}) + \left\langle \mathbf{w} - \mathbf{w}_{t+1}, \mathbf{g}_{t+1} \right\rangle\right\} \tag{2.21}$$

where $\mathbf{g}_{t+1} \in \partial J(\mathbf{w}_{t+1})$. (See Figure 2.3 for an illustration of the construction of piecewise linear lower bound of a 1-dimensional convex function.) The iteration repeats until the approximation gap

$$\epsilon_t := \min_{i \in [t+1]} J(\mathbf{w}_i) - \check{J}_t(\mathbf{w}_{t+1}) \tag{2.22}$$

is smaller than a pre-defined precision $\epsilon > 0$. Since $J(\mathbf{w}) \geq \check{J}_t(\mathbf{w})\ \forall\, \mathbf{w}, \forall t > 0$, it follows that

$$\min_{\mathbf{w} \in U} J(\mathbf{w}) \geq \min_{\mathbf{w} \in U} \check{J}_t(\mathbf{w})$$

for all $t \geq 1$. Thus, the approximation gap (2.22) is in fact an upper bound on the gap (2.17). See Figure 2.4 for an illustration of approximation gap.

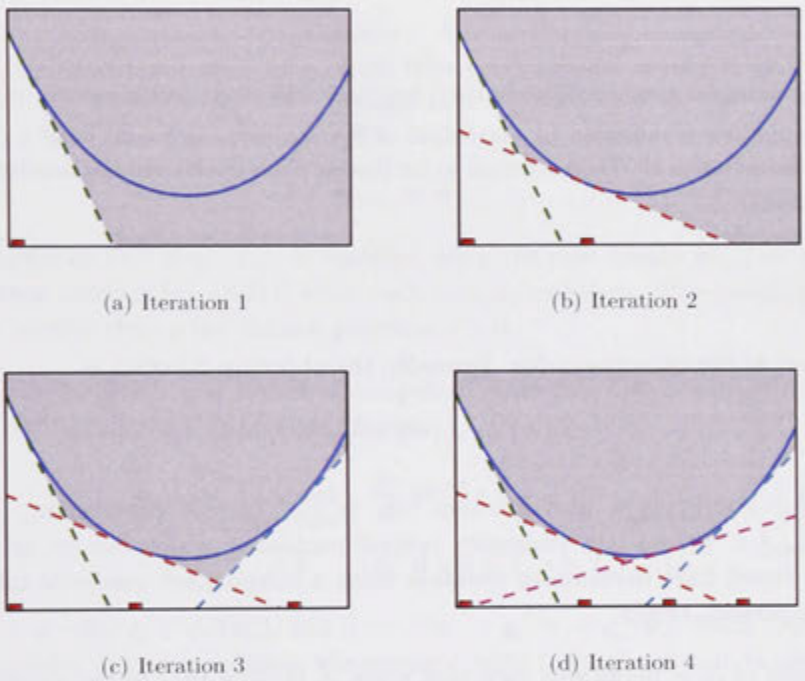
Although Kelly's cutting-plane method has been shown to converge globally [Kelly, 1960], there is no rate of convergence given. It fact, the cutting-plane method can take a large number of iterations to converge in some cases [Mäkelä, 2002, and references therein].

Since (2.20) may have non-unique solutions especially when $d > t$, it is likely that the iterates generated and their linearizations are systematically making little or no improvement to the approximation $\check{J}_t$. To illustrate this phenomena, Example XV.1.1.2 in Hiriart-Urruty and Lemaréchal [1993], credited to Nemirovskii [Belloni, 2005], shows that for any precision $\epsilon \in (0, 1)$, there always exists a function for which the cutting-plane method will require $O(\epsilon^{-d/2})$ iterations to find an ϵ-accurate solution.

In addition to large number of iterations, cutting-plane method also suffers from com-

putation and memory issues as the number of constraints in the LP grows with the number of iterations.

Numerous methods have been proposed to improve the convergence speed of Kelly's cutting-plane method, namely, Center of Gravity, Largest Inscribed Sphere, Volumetric, and Analytic Center. Since these methods are not the central theme of this thesis, we do not discuss them here but refer the interested reader to Elhedhli et al. [2008] for a brief overview and the references therein for more detailed descriptions.



(a) Iteration 1 (b) Iteration 2

(c) Iteration 3 (d) Iteration 4

Figure 2.3: A convex function (blue solid curve) is bounded from below by its linearizations (dashed lines). The gray area indicates the piecewise linear lower bound obtained by using the linearizations. We depict a few iterations of the cutting-plane method. At each iteration the piecewise linear lower bound is minimized and a new linearization is added at the minimizer (red rectangle). As can be seen, adding more linearizations improves the lower bound.

2.3.3 Proximal Bundle Method

To prevent cutting-plane method from taking long steps which cause instability and slow convergence [Hiriart-Urruty and Lemaréchal, 1993], the proximal bundle method [Kiwiel, 1990, Lemaréchal, 1978] enforces a proximity control to the iterate update such that the new iterate stays sufficiently close to previous best iterate and yields a

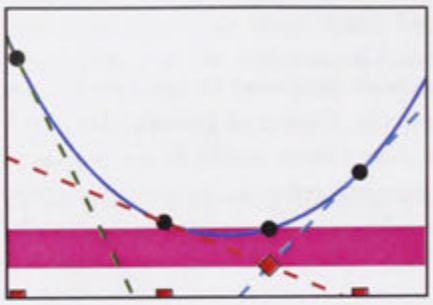


Figure 2.4: A convex function (blue solid curve) with three linearizations (dashed lines) evaluated at three different locations (red squares). The approximation gap ϵ_3 at the end of third iteration is indicated by the height of the magenta horizontal band *i.e.*, difference between lowest value of $J(\mathbf{w})$ evaluated so far (lowest black circle) and the minimum of $\check{J}_3(\mathbf{w})$ (red diamond).

decrement in the objective value. Formally, the objective function is

$$\check{J}_t(\mathbf{w}) := \tilde{J}_t(\mathbf{w}) + \frac{\mu_t}{2} \|\mathbf{w} - \hat{\mathbf{w}}_t\|_2^2 ,$$

where $\frac{1}{2}\|\cdot - \hat{\mathbf{w}}_t\|_2^2$ is a prox-function, $\hat{\mathbf{w}}_t$ is the current prox-center, and $\mu_t \in [\mu_{\min}, \mu_{\max}] \subset (0, \infty)$ is a proximity control parameter which can be set to a constant or tuned from iteration to iteration using a safeguarded quadratic interpolation strategy [Kiwiel, 1990].

The tuning of μ_t is performed such that when $\tilde{J}_t$ is *close* to J in the vicinity of prox-center, the new iterate should not be restricted from staying far from prox-center; this is achieved by making the prox-function less dominant *i.e.*, setting a smaller value to μ_t. On the contrary, μ_t is set to a larger value so that prox-function is more dominant and the new iterate will stay close to prox-center when J is not well-approximated by $\tilde{J}_t$. Although Kiwiel [1990] showed that the strategy improves the convergence compared to that with μ_t fixed at 1, this strategy has at least three parameters which may require careful tuning.

Unlike cutting-plane method, the iterate update rule of proximal bundle method

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \check{J}_t(\mathbf{w}). \tag{2.23}$$

is not restricted to a compact set as the strongly convex (see Definition C.0.7) prox-function ensures the existence of a unique minimizer for J_t. Furthermore, the prox-function prevents the new iterate from moving too far from the current prox-center

which is the best estimate of the minimizer obtained so far. A *serious* step

$$\hat{\mathbf{w}}_{t+1} := \mathbf{w}_{t+1}$$

is taken to replace the new prox-center $\hat{\mathbf{w}}_{t+1}$ to the new iterate $\mathbf{w}_{t+1}$ if $\mathbf{w}_{t+1}$ is significantly better than $\hat{\mathbf{w}}_t$, *i.e.*,

$$J(\hat{\mathbf{w}}_t) - J(\mathbf{w}_{t+1}) \geq \rho\epsilon_t, \tag{2.24}$$

where $\rho \in (0, 1)$ is a descent parameter and

$$\epsilon_t := J(\hat{\mathbf{w}}_t) - \check{J}_t(\mathbf{w}_{t+1})$$

is the approximation gap. If (2.24) does not hold, the prox-center remains the same, *i.e.*,

$$\hat{\mathbf{w}}_{t+1} := \hat{\mathbf{w}}_t.$$

This is known as *null* step. $\check{J}_{t+1}$ is updated using the new iterate $\mathbf{w}_{t+1}$ as is done in cutting-plane method (*cf.* (2.21)) after each serious/null step. The iteration repeats until ϵ_t is smaller than a pre-defined precision $\epsilon > 0$.

We now describe how a new iterate is computed. Note that subproblem (2.23) can be rewritten as the following quadratic program:

$$\mathbf{d}_{t+1} = \underset{\xi \in \mathbb{R},\ \mathbf{d} \in \mathbb{R}^d}{\operatorname{argmin}} \quad \frac{\mu_t}{2} \|\mathbf{d}\|_2^2 + \xi \tag{2.25a}$$

$$\text{s.t.} \quad \mathbf{g}_i^\top \mathbf{d} + b_i \leq \xi,\ i = 1, \ldots, t \tag{2.25b}$$

where $\mathbf{d} := \mathbf{w} - \hat{\mathbf{w}}_t$, $\mathbf{g}_i \in \partial J(\mathbf{w}_i)$, and $b_i := J(\mathbf{w}_i) - \mathbf{g}_i^\top \mathbf{w}_i + \mathbf{g}_i^\top \hat{\mathbf{w}}_t$. Since (2.25) is convex and satisfies Slater's condition, the strong duality holds [Boyd and Vandenberghe, 2004]. Therefore, (2.25) can be solved exactly via its Lagrange dual:

$$\boldsymbol{\alpha}_t = \underset{\boldsymbol{\alpha} \in \mathbb{R}^t}{\operatorname{argmax}} \quad -\frac{1}{2\mu_t} \boldsymbol{\alpha}^\top \mathbf{Q}_t \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{b}_t \tag{2.26a}$$

$$\text{s.t.} \quad \boldsymbol{\alpha}^\top \mathbf{1}_t = 1,\ \boldsymbol{\alpha} \geq 0. \tag{2.26b}$$

where $\mathbf{Q}_t = \mathbf{G}_t^\top \mathbf{G}_t$ with $\mathbf{G}_t := [\mathbf{g}_1, \ldots, \mathbf{g}_t]$ a matrix of columns[6] of subgradients $\mathbf{g}_i$, $\mathbf{b}_t := (b_1, \ldots, b_t)$ is a vector offsets b_i, and $\mathbf{1}_t$ is a t-dimensional vector with all components being one. By the connection of primal and dual variables, *i.e.*,

$$\mathbf{d}_{t+1} = -\mu_t^{-1} \mathbf{G}_t \boldsymbol{\alpha}_t,$$

[6]Subgradients $\mathbf{g}_i$ are assumed to be column vectors.

we obtain the new iterate

$$\mathbf{w}_{t+1} := \hat{\mathbf{w}}_t - \mu_t^{-1}\, \mathbf{G}_t\, \boldsymbol{\alpha}_t.$$

For the case where $d \gg t$, the Lagrange dual (2.26) that has t variables is relatively easier to solve compared to its primal problem (2.25) which has d variables.

We also note that the Hessian matrix $\mathbf{Q}_t$ of (2.26a) is expanded by only one row and one column at each iteration *i.e.*,

$$\mathbf{Q}_{t+1} = \begin{pmatrix} \mathbf{Q}_t & \mathbf{G}_t^\top\, \mathbf{g}_{t+1} \\ \mathbf{g}_{t+1}^\top\, \mathbf{G}_t & \mathbf{g}_{t+1}^\top\, \mathbf{g}_{t+1} \end{pmatrix}$$

with $\mathbf{G}_{t+1} = [\mathbf{G}_t, \mathbf{g}_{t+1}]$. This structural property is exploited to speed up the QP computation time by special active-set QP solvers [Kiwiel, 1989, Frangioni, 1997] that solves a QP of type (2.26) by solving its associated Karush-Kuhn-Tucker (KKT) system [see *e.g.*, Vanderbei, 2008]. In these solvers, a lower/upper trapezoidal factorization [see *e.g.*, Golub and Van Loan, 1996] of $\mathbf{Q}_t$ is kept and updated for each addition/deletion of linearizations in $\check{J}_t$. The time complexity for updating the factorization and for solving the KKT system are $O(dt)$ and $O(t^2)$, respectively [Kiwiel, 1989]. This is considerably more efficient than standard methods for solving the KKT system (*i.e.*, system of linear equations) such as the conjugate gradient method which also takes $O(dt)$ time to update $\mathbf{Q}_t$ and $O(t^\beta)$, $\beta \in (2, 3]$ time to solve the system [Nocedal and Wright, 1999].

Memory and computational issues arise as t increases because $\check{J}_t$ stores a bundle of t linearizations in $O(dt)$ space and the dimension of QP becomes larger (hence requires longer time to solve). To overcome these implementation issues, Kiwiel [1983, 1985] proposed two strategies, namely, linearization aggregation and linearization selection to reduce the number of linearizations in $\check{J}_t$. The aggregation strategy combines two or more linearizations $\bar{J}_{\mathbf{w}_i}$, $i \in I \subset \{1, \ldots, t\}$ into a new aggregate linearization where its corresponding aggregate subgradient, offset, and dual variable are defined as:

$$\tilde{\mathbf{g}}^I := \frac{1}{\tilde{\alpha}^I} \sum_{i \in I} \alpha_i\, \mathbf{g}_i, \quad \tilde{b}^I := \frac{1}{\tilde{\alpha}^I} \sum_{i \in I} \alpha_i b_i, \quad \text{and} \quad \tilde{\alpha}^I := \sum_{i \in I} \alpha_i,$$

respectively. Then, the elements $\mathbf{g}_i$, b_i, and α_i, for all $i \in I$ are replaced by their respective aggregates. For notational convenience, $\tilde{\mathbf{g}}^I$, $\tilde{b}^I$, and $\tilde{\alpha}^I$ can be renamed to $\mathbf{g}_i$, b_i, and α_i for any $i \in I$. This strategy guarantees (dual) feasibility of the solution of (2.26) as the new set of dual variables still satisfies the simplex constraint (*i.e.*, variables are non-negative and sum up to one).

The linearization selection strategy does not bound the number of linearizations which the aggregation strategy does, but is more natural and simpler to implement: After each iteration, the linearizations with corresponding dual variables being zero are removed from the bundle. In addition to the feasibility, this strategy also guarantees the

optimality of (2.26). Clearly, both strategies can be employed simultaneously without conflict.

Algorithm 1 provides details about the proximal bundle method we have just described. Kiwiel [2000] proved that proximal bundle method converges to ϵ-accurate solution in

Algorithm 1 Proximal Bundle Method

input : $\epsilon > 0$, $\rho \in (0,1)$, $\mu_1 \in [\mu_{\min}, \mu_{\max}]$, $\mathbf{w}_1$
initialization: $t \leftarrow 1$, $\hat{\mathbf{w}}_1 \leftarrow \mathbf{w}_1$, $B = \{1\}$
loop
  Compute $J(\mathbf{w}_t)$ and $\mathbf{g}_t \in \partial J(\mathbf{w}_t)$
  Update $\check{J}_t(\mathbf{w}) := \max_{i \in B}\{J(\mathbf{w}_i) + \langle \mathbf{w} - \mathbf{w}_i, \mathbf{g}_i \rangle\}$
  $\mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w}} \check{J}_t(\mathbf{w}) + \frac{\mu_t}{2} \|\mathbf{w} - \hat{\mathbf{w}}_t\|_2^2$
  $\epsilon_t \leftarrow J(\hat{\mathbf{w}}_t) - \check{J}_t(\mathbf{w}_{t+1})$
  if $\epsilon_t < \epsilon$ **then**
    return $\hat{\mathbf{w}}_t$
  end if
  if $J(\hat{\mathbf{w}}_t) - J(\mathbf{w}_{t+1}) \geq \rho \epsilon_t$ **then**
    SERIOUS STEP: $\hat{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_{t+1}$
  else
    NULL STEP: $\hat{\mathbf{w}}_{t+1} \leftarrow \hat{\mathbf{w}}_t$
  end if
  $B \leftarrow B \cup \{t+1\}$
  [Optional] $\mu_{t+1} \leftarrow \mu_t$ or update $\mu_{t+1} \in [\mu_{\min}, \mu_{\max}]$ according to Kiwiel [1990]
  [Optional] Perform linearizations aggregation or selection (see text) and update index set B accordingly
  $t \leftarrow t + 1$
end loop

$O(\epsilon^{-3})$ steps.

Robinson [1999] showed that the number of serious steps in proximal bundle method is of order $O(\log(\epsilon^{-1}))$ for strongly convex J (*e.g.*, with a strongly convex regularizer such as L_2 norm). Also, the analysis of [Kiwiel, 2000] shows that the number of null steps between two adjacent serious steps is of order $O(\epsilon^{-1})$. Combining the two results, we show in Theorem 2.3.1 that Algorithm 1 converges to ϵ-accurate solution in $O(\epsilon^{-1} \log(\epsilon^{-1}))$ steps when J is strongly convex.

Let us first define $\delta_t := J(\hat{\mathbf{w}}_t) - \mathring{J}_t(\mathbf{w}_{t+1})$ and let c_t be the Lipschitz constant of J and $\check{J}_t$ in the neighborhood $U_t := \{\mathbf{w} \ : \ \|\mathbf{w} - \hat{\mathbf{w}}_t\| \leq \delta_t / \mu_{\min}\}$.

Theorem 2.3.1. *Assume $J : \mathbb{R}^d \to \mathbb{R}$ is a σ-strongly convex function. Algorithm 1 finds an iterate $\mathbf{w}_T$ after*

$$T \leq \left[1 + \frac{4c^2}{\epsilon \mu_{\min}(1-\rho)^2}\right] \log\left(\frac{\epsilon_1(1 + \mu_{\max}\sigma^{-1})}{\epsilon(1-\rho)}\right)$$

steps where $c := \max_{i \in [T]} c_i$, *such that* $J(\mathbf{w}_T) \leq \min_{\mathbf{w}} J(\mathbf{w}) + \epsilon$, *for any* $\epsilon > 0$.

A proof sketch for this theorem can be found in Appendix B.2.

Amongst other interesting developments in the area we mention two. Kiwiel [1999] proposed an extension of proximal bundle method by replacing the prox-function $\frac{1}{2}\|\mathbf{w} - \hat{\mathbf{w}}_t\|_2^2$ with a Bregman distance [Bregman, 1967] $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined as:

$$D_h(\mathbf{w}, \hat{\mathbf{w}}_t) = h(\mathbf{w}) - h(\hat{\mathbf{w}}_t) - \langle \mathbf{w} - \hat{\mathbf{w}}_t, \nabla h(\hat{\mathbf{w}}_t) \rangle \,,$$

where $h : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable strictly convex function and $\nabla h(\mathbf{u})$ denotes the gradient of h at $\mathbf{u}$. For $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, D_h is exactly the prox-function used in proximal bundle method. Frangioni [2002] also proposed to replace the prox-function with a general class of convex function $D : \mathbb{R}^d \to \mathbb{R}$ taking the direction $\mathbf{d} := \mathbf{w} - \hat{\mathbf{w}}_t$ as the only argument. (In general, the methods of Kiwiel [1999] and Frangioni [2002] do not generalize each other.)

2.3.4 Trust-Region Bundle Method

The trust-region bundle method introduced by Schramm and Zowe [1992] is a variant of proximal bundle method (see Section 2.3.3). It uses different strategy for updating the proximity control weight μ_t and different conditions for detecting a serious step.

In particular, the update of μ_t is motivated by the trust region philosophy [Conn et al., 2000]. To see how trust-region idea is applicable in this case, we rewrite the subproblem (2.23) of proximal bundle method as the following equivalent formulation:

$$\mathbf{w}_{t+1} := \operatorname*{argmin}_{\mathbf{w}} \left\{ \check{J}_t(\mathbf{w}) \;:\; \|\mathbf{w} - \hat{\mathbf{w}}_t\| \leq \tau_t \right\}$$

where $\tau_t > 0$ is inversely proportional to μ_t [Schramm and Zowe, 1992] (*cf.* Section 2.1.3). In addition to the serious step condition (2.24) in proximal bundle method, trust-region bundle method further requires that the model is substantially changed, *i.e.*, $\check{J}_{t+1} > \check{J}_t$, before a serious step is taken. If only the latter condition is not satisfied, the trust-region parameter is increased (*i.e.*, $\tau_{t+1} > \tau_t$) so that the new prox-center can possibly be found in an enlarged region around current prox-center. Otherwise, a null step is taken. During a null step, if the serious step condition is missed by a large value, the trust-region parameter will be shrunk to confine the search of new iterate in a smaller region around current prox-center.

Despite of the trust-region principle and formulation used in developing the scheme for updating τ_t, the actual implementation of trust-region bundle method is exactly the same as that of proximal bundle method [Schramm and Zowe, 1992]. This is because the reciprocal of τ_t can be taken as μ_t in proximal bundle method.

Schramm and Zowe [1992] showed that trust-region bundle method converges in finite number of steps using the analysis of Kiwiel [1985]. Later, Kiwiel [2000] pointed out that the convergence rate of this method is of order $O(\epsilon^{-3})$. In fact, when J is strongly convex, Theorem 2.3.1 also applies to this method as its algorithm differs from that of proximal bundle method only in the update of proximity control parameter μ_t which can always be safely fixed to constant.

2.3.5 Level-Set Bundle Method

The level-set bundle method developed by Lemaréchal et al. [1995] is a variant of proximal bundle method that directly controls the value of the model $\check{J}_t(\mathbf{w}_{t+1})$ when optimizing for new iterate $\mathbf{w}_{t+1}$. This is different from proximal and trust-region bundle methods where the model value $\check{J}_t(\mathbf{w}_{t+1})$ is indirectly determined by the parameters μ_t and τ_t, respectively.

The resulting iterate update rule reads:

$$\mathbf{w}_{t+1} := \operatorname*{argmin}_{\mathbf{w} \in U} \tfrac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 \quad \text{subject to} \quad \check{J}_t(\mathbf{w}) \leq \zeta_t,$$

where $U \subset \mathbb{R}^d$ is assumed to be bounded,

$$\zeta_t := \gamma v^t + (1 - \gamma) v_t = v_t + \gamma \epsilon_t$$

is a level-set parameter, $\gamma \in (0, 1)$ is a fixed parameter, $v^t := \min_{1 \leq i \leq t} J(\mathbf{w}_i)$, $v_t := \min_{\mathbf{w} \in U} \check{J}_t(\mathbf{w})$, and $\epsilon_t := v^t - v_t$ is the current approximation gap.

In addition to the update rule, Lemaréchal et al. [1995] also suggested to make every iteration a serious step as the level-sets of the model is claimed to be rather stable. These modifications substantially simplify the algorithm, compared to proximal and trust-region bundle methods. The convergence rate was shown to be of order $O(\epsilon^{-2})$ which is an order of magnitude better than proximal and trust-region bundle methods due to the boundedness of U [Kiwiel, 2000].

Despite of its better convergence rate, the per iteration computation of level-set bundle method is more expensive than that of proximal and trust-region bundle methods: At each iteration a linear program solver (*cf.* cutting-plane method) is called to compute the exact minimum of $\check{J}_t$ *i.e.*, v_t. Then a quadratic program solver is called in order to compute the new iterate.

2.3.6 Variable Metric Bundle Method

The proximal, trust-region and level-set bundle methods we have described can be characterized as having a quadratic form prox-function D *i.e.*,

$$D(\mathbf{w}, \hat{\mathbf{w}}_t) = (\mathbf{w} - \hat{\mathbf{w}}_t)^\top \mathbf{M}(\mathbf{w} - \hat{\mathbf{w}}_t).$$

where $\mathbf{M}$ is a $d \times d$ identity matrix scaled by a factor $\mu \in (0, \infty)$. The prox-function can be, in turn, regarded as a metric measuring the "distance" between points $\mathbf{w}$ and $\hat{\mathbf{w}}_t$. As μ changes, the scale of all the coordinates of $\mathbb{R}^d$ is changed accordingly. Since the metric D here is fully determined by μ which changes from iteration to iteration, these methods are also called diagonal variable metric bundle method [Mäkelä, 2002].

Lemaréchal [1978] presented an algorithm of the family where $\mathbf{M}$ is any positive semi-definite matrix. The role of $\mathbf{M}$ is to model the second order derivative of the objective J using only first order information of J *i.e.*, (sub)gradients. Similar to the classical BFGS method for unconstrained differentiable objective [Nocedal and Wright, 1999], $\mathbf{M}$ is updated by a secant formula at every iteration.

According to Mäkelä [2002], this variable metric bundle method by Lemaréchal [1978] did not gain much popularity partly due to the poor numerical performance reported in Lemaréchal [1982]. Lemaréchal and Sagastizábal [1997] developed a related method which employs a "reversal" quasi-Newton formula for updating $\mathbf{M}$ and uses a special curved line search procedure to find a step size in computing the new iterate. The numerical experiments in Lemaréchal and Sagastizábal [1997] shows that this method is comparable to proximal, trust-region, and level-set methods.

Vlček and Lukšan [1999] proposed another variable metric bundle method which uses the secant formula of BFGS to update $\mathbf{M}^{-1}$ *i.e.*, the inverse of $\mathbf{M}$ and solves a quadratic program with only three dual variables at every iteration: Two of the dual variables correspond to the linearizations computed at current prox-center and new iterate, and the other dual variable is for the aggregate linearization [Kiwiel, 1990].

Although the variable metric provides more (*i.e.*, second order) information about the objective, it takes d^2 floating-point numbers to store $\mathbf{M}$ or $\mathbf{M}^{-1}$ and hence is infeasible when d is large. Haarala et al. [2007] tackled this problem by taking the approach similar to what limited memory variable metric methods (*e.g.*, L-BFGS) apply to standard variable metric methods (*e.g.*, BFGS) (see Nocedal and Wright [1999] for details). Similar to Vlček and Lukšan [1999] the limited memory bundle method of Haarala et al. [2007] solves a three variables quadratic program at every iteration. However, unlike Vlček and Lukšan [1999], the matrix $\mathbf{M}^{-1}$ is never stored explicitly. This is because, in the algorithm, $\mathbf{M}^{-1}$ involves only in matrix-vector multiplications. Therefore, $\mathbf{M}^{-1}$ is explicitly represented in its primitive form *i.e.*, two k-by-d matrices and the matrix-vector multiplication is done by using standard limited memory BFGS update formula [see *e.g.*, Nocedal and Wright, 1999, Chapter 7]. The memory saving

here is due to the user-specified number k which is usually much smaller than d.

2.4 Summary

In this chapter, we reviewed the regularized risk minimization framework which helps turning learning problems into well-studied convex optimization problems. We also reviewed many state-of-the-art machine learning algorithms for solving the convex regularized risks.

The bundle methods is of particular interest here because it allows one to treat the regularized risk as a black box function, regardless of its differentiability. In addition to being general, bundle methods, as a batch solver, are easily amenable to the parallel and distributed computation setting. This, in general, does not hold for other general solvers such as the online subgradient methods.

In the following chapters, we develop a bundle method specialized to the case of regularized risk minimization. We prove that the convergence rate of the proposed method is of order $O(\epsilon^{-1})$ for non-differentiable empirical risk and $O(\log(\epsilon^{-1}))$ for differentiable empirical risk, with a strongly convex regularizer. These rates are much better than the $O(\epsilon^{-1}\log(\epsilon^{-1}))$ rate we have shown in Section 2.3.3. Furthermore, the proposed method allows efficient computation of regularization path and model selection simply by reusing the approximation of empirical risk repeatedly.

Chapter 3

Bundle Methods for Regularized Risk Minimization (BMRM)

In this chapter, we develop a bundle method specialized to minimizing the regularized risk

$$J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w}).$$

Our method, BMRM, differs from the standard bundle methods (see Section 2.3) in four aspects:

- BMRM neither maintains a prox-center nor follows the concept of serious/null steps, hence the algorithm is simpler. Furthermore, BMRM has no algorithm specific parameters which usually require careful tuning;
- BMRM sacrifices the generality of bundle methods a little by knowing (only) that the regularized risk $J(\mathbf{w})$ is a sum of two convex functions, namely an easy to evaluate regularizer $\Omega(\mathbf{w})$ and a difficult to evaluate empirical risk $R(\mathbf{w})$;
- BMRM does not augment the regularized risk with an additional strongly convex prox-function; instead, it uses $\Omega(\mathbf{w})$;
- BMRM builds approximation (*i.e.*, piecewise linear lower bound) only for the empirical risk instead of the regularized risk.

Nevertheless, BMRM retains many other useful properties of standard bundle methods such as the linearization selection and aggregation strategies for controlling the bundle size.

In this chapter, we first describe the basic BMRM algorithm and its variants. Then we provide convergence analysis which shows that BMRM has a better convergence rate than its closest match *i.e.*, the proximal bundle method when the regularizer is strongly

convex. We also describe the implementation of BMRM, covering issues such as memory efficiency and parallel/distributed computation. Following that, we demonstrate in experiments the convergence behavior of BMRM under various computational and learning conditions.

3.1 Algorithms

The algorithm of BMRM is similar but simpler compared to the proximal bundle method. At t-th iteration, the value $R(\mathbf{w}_t)$ and a subgradient $\mathbf{a}_t \in \partial R(\mathbf{w}_t)$ are evaluated at the current iterate $\mathbf{w}_t$. With the subgradient $\mathbf{a}_t$ and offset $b_t := R(\mathbf{w}_t) - \langle \mathbf{w}_t, \mathbf{a}_t \rangle$, BMRM builds the piecewise linear lower bounding approximation $\check{R}_t$ of R as follows:

$$\check{R}_t(\mathbf{w}) := \max_{i \in [t]} \{ \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i \} .$$

Then BMRM computes the new iterate $\mathbf{w}_{t+1}$ by

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \left\{ J_t(\mathbf{w}) := \lambda \Omega(\mathbf{w}) + \check{R}_t(\mathbf{w}) \right\} \tag{3.1}$$

where J_t is a piecewise convex lower bound for J. The algorithm terminates when the approximation gap

$$\epsilon_t := \min_{i \in [t]} J(\mathbf{w}_i) - J_t(\mathbf{w}_{t+1})$$

is less than or equal to a pre-defined accuracy $\epsilon > 0$. Algorithm 2 lists the details of the procedure we have just described.

Algorithm 2 BMRM

1: **input:** $\epsilon > 0$, $\mathbf{w}_1 \in \mathbb{R}^d$
2: **initialization:** $t \leftarrow 0$
3: **repeat**
4: $\quad t \leftarrow t + 1$
5: $\quad$ Compute $\mathbf{a}_t \in \partial R(\mathbf{w}_t)$ and $b_t \leftarrow R(\mathbf{w}_t) - \langle \mathbf{w}_t, \mathbf{a}_t \rangle$
6: $\quad$ Update $\check{R}_t(\mathbf{w}) := \max_{i \in [t]} \{ \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i \}$
7: $\quad \mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w}} J_t(\mathbf{w}) := \lambda \Omega(\mathbf{w}) + \check{R}_t(\mathbf{w})$
8: $\quad \mathbf{w}_* := \operatorname{argmin}_{k \in [t+1]} J(\mathbf{w}_k)$
9: $\quad \epsilon_t \leftarrow J(\mathbf{w}_*) - J_t(\mathbf{w}_{t+1})$
10: **until** $\epsilon_t \leq \epsilon$
11: **return** $\mathbf{w}_*$

The crux of the algorithm is in solving the subproblem (3.1). Similar to the proximal bundle method, we resort to the dual problem of (3.1). We derive the dual problems corresponding to various common choice of regularizers in the following section.

3.1.1 The Dual of Subproblem (3.1)

Let us first define the Fenchel dual or *conjugate* of the regularizer Ω.

Definition 3.1.1 (Fenchel Dual). *Denote by $\Omega : U \to \mathbb{R} \cup \{+\infty\}$ a convex function on a convex set U not identically $+\infty$. Then the dual Ω^* of Ω is defined as*

$$\Omega^*(\boldsymbol{\mu}) := \sup \{\langle \mathbf{w}, \boldsymbol{\mu} \rangle - \Omega(\mathbf{w}) \ : \ \mathbf{w} \in \operatorname{dom} \Omega\}. \tag{3.2}$$

Several choices of regularizers are common in machine learning applications. For $U = \mathbb{R}^d$ the squared L_2 norm regularizer yields

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2 \qquad \text{and} \qquad \Omega^*(\boldsymbol{\mu}) = \frac{1}{2} \|\boldsymbol{\mu}\|_2^2.$$

More generally, for L_p norms one obtains [Boyd and Vandenberghe, 2004, Shalev-Shwartz and Singer, 2006]:

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_p^2 \qquad \text{and} \qquad \Omega^*(\boldsymbol{\mu}) = \frac{1}{2} \|\boldsymbol{\mu}\|_q^2 \text{ where } \frac{1}{p} + \frac{1}{q} = 1.$$

For any symmetric positive definite matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ and any $\mathbf{v} \in \mathbb{R}^d$, we have a quadratic form regularizer and its dual [Hiriart-Urruty and Lemaréchal, 1993, Example X.1.1.3] as:

$$\Omega(\mathbf{w}) = \frac{1}{2} \mathbf{w}^\top \mathbf{M} \mathbf{w} + \langle \mathbf{v}, \mathbf{w} \rangle \qquad \text{and} \qquad \Omega^*(\boldsymbol{\mu}) = \frac{1}{2} (\boldsymbol{\mu} - \mathbf{v})^\top \mathbf{M}^{-1} (\boldsymbol{\mu} - \mathbf{v}).$$

For the *unnormalized* negative entropy, where $U = \mathbb{R}_+^d$, we have

$$\Omega(\mathbf{w}) = \sum_i w_i \log w_i \qquad \text{and} \qquad \Omega^*(\boldsymbol{\mu}) = \sum_i \exp \mu_i.$$

For the *normalized* negative entropy, where $U = \{\mathbf{w} \mid \mathbf{w} \geq 0 \text{ and } \|\mathbf{w}\|_1 = 1\}$ is the probability simplex, we have

$$\Omega(\mathbf{w}) = \sum_i w_i \log w_i \qquad \text{and} \qquad \Omega^*(\boldsymbol{\mu}) = \log \sum_i \exp \mu_i.$$

In general, any convex regularizer $\Omega : \mathbb{R}^d \to \mathbb{R}$ can be modified to enforce convex constraints on $\mathbf{w}$ by restricting its domain to a closed convex set U, *i.e.*,

$$\Omega_U(\mathbf{w}) := \Omega(\mathbf{w}) + \mathbf{i}_U(\mathbf{w}), \quad \text{where } \mathbf{i}_U(\mathbf{w}) = \begin{cases} 0 & \text{if } \mathbf{w} \in U, \\ \infty & \text{otherwise} \end{cases}.$$

Examples of U commonly seen in machine learning applications include the

$$\text{hypercube:} \quad \{\mathbf{w} \;:\; \mathbf{w} \in [p_1, q_1] \times \cdots \times [p_d, q_d],\ p_i \leq q_i\ \forall i \in [d]\}\,, \text{ and the}$$
$$\text{polyhedron:} \quad \left\{\mathbf{w} \;:\; \mathbf{H}\,\mathbf{w} \leq \mathbf{v},\ \mathbf{H} \in \mathbb{R}^{d \times d},\ \mathbf{v} \in \mathbb{R}^d\right\},$$

and the combination of both. However, the resulting dual Ω_U^* may still remain in the original variational form (3.2) instead of the analytic form which is generally easier to handle.

We now state the dual problem of (3.1) in the following theorem.

Theorem 3.1.2. *Denote by* $\mathbf{A}_t = [\mathbf{a}_1, \ldots, \mathbf{a}_t]$ *the matrix whose columns are the (sub)gradients, and let* $\mathbf{b}_t = [b_1, \ldots, b_t]$. *The dual problem of*

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \{J_t(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + \max_{i \in [t]} \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i\} \quad \textit{is} \tag{3.3}$$

$$\boldsymbol{\alpha}_t = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathbb{R}^t} \{J_t^*(\boldsymbol{\alpha}) := -\lambda\Omega^*(-\lambda^{-1}\,\mathbf{A}_t\,\boldsymbol{\alpha}) + \boldsymbol{\alpha}^\top\,\mathbf{b}_t \;\mid\; \boldsymbol{\alpha} \geq 0_t,\ \|\boldsymbol{\alpha}\|_1 = 1\}. \tag{3.4}$$

Furthermore, $\mathbf{w}_{t+1}$ *and* $\boldsymbol{\alpha}_t$ *are related by the dual connection* $\mathbf{w}_{t+1} = \partial\Omega^*(-\lambda^{-1}\,\mathbf{A}_t\,\boldsymbol{\alpha}_t)$.

Proof. We rewrite (3.3) as a constrained optimization problem: $\min_{\mathbf{w},\xi} \lambda\Omega(\mathbf{w}) + \xi$ subject to $\xi \geq \langle \mathbf{w}, \mathbf{a}_i \rangle + b_i$ for $i = 1, \ldots, t$. The corresponding Lagrange function can be written as

$$L(\mathbf{w}, \xi, \boldsymbol{\alpha}) = \lambda\Omega(\mathbf{w}) + \xi - \boldsymbol{\alpha}^\top \left(\xi \mathbf{1}_t - \mathbf{A}_t^\top\,\mathbf{w} - \mathbf{b}_t\right) \text{ with } \boldsymbol{\alpha} \geq 0_t, \tag{3.5}$$

Taking derivatives with respect to ξ yields $1 - \boldsymbol{\alpha}^\top \mathbf{1}_t = 0$. Moreover, minimization of L with respect to $\mathbf{w}$ implies solving $\max_{\mathbf{w}} \left\langle \mathbf{w}, -\lambda^{-1}\,\mathbf{A}_t\,\boldsymbol{\alpha} \right\rangle - \Omega(\mathbf{w}) = \Omega^*(-\lambda^{-1}\,\mathbf{A}_t\,\boldsymbol{\alpha})$. Substituting both terms back into (3.5) allows us to eliminate the primal variables ξ and $\mathbf{w}$. □

If Ω is strictly convex, then Ω^* is continuously differentiable on the interior of dom Ω^* (see Definition C.0.10). So, for differentiable Ω^*, many well known and efficient constrained smooth optimization techniques [Nocedal and Wright, 1999] are readily applicable for solving the dual problem (3.4). Under a stronger condition where Ω is strongly convex, we have the property that the dual Ω^* has Lipschitz continuous gradient (see Definition C.0.8); a property that we will make use of in proving the convergence of BMRM. We note that all the regularizers we discussed above are strongly convex (including their modifications with constraints).

We state here an immediate corollary for $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, which is a very commonly used regularizer in machine learning applications.

Corollary 3.1.3. *For squared* L_2 *norm regularization,* i.e., $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, (3.4)

becomes

$$\boldsymbol{\alpha}_t = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathbb{R}^t} \{ -\tfrac{1}{2\lambda} \boldsymbol{\alpha}^\top \mathbf{A}_t^\top \mathbf{A}_t \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{b}_t \mid \boldsymbol{\alpha} \geq \mathbf{0}_t, \ \|\boldsymbol{\alpha}\|_1 = 1 \}.$$

This means that the dual problem is a QP hence the efficient QP solvers [Kiwiel, 1989, Frangioni, 1997] developed for standard bundle methods can be used in this case. In fact, we do not even need to know the (sub)gradients explicitly. All that is required to define the QP are the inner products between the (sub)gradients $\langle \mathbf{a}_i, \mathbf{a}_j \rangle$.

3.1.2 BMRM with Line Search

The algorithm of BMRM can, in addition, employ a line search to speed up the convergence. We present here one such variant that generalizes the optimized cutting plane algorithm for support vector machines (OCAS) of Franc and Sonnenburg [2008]. This variant first builds $\bar{R}_t$ and minimizes J_t to obtain an intermediate iterate $\mathbf{w}_{t+1}$. Then, it performs a line search along the line joining the *best* iterate $\mathbf{w}_t^b$ and the intermediate iterate $\mathbf{w}_{t+1}$ to obtain a new best iterate $\mathbf{w}_{t+1}^b$ which acts like the prox-center in the proximal bundle method. Since $\mathbf{w}_{t+1} - \mathbf{w}_t^b$ is not necessarily a direction of descent, the line search might return a zero step.

Instead of using $\mathbf{w}_{t+1}^b$ as the new iterate for generating new linearization, Franc and Sonnenburg [2008] proposed to use a pre-set parameter $\theta \in (0, 1]$ to generate a *cut* iterate $\mathbf{w}_{t+1}^c$ on the line segment joining $\mathbf{w}_{t+1}^b$ and $\mathbf{w}_{t+1}$. Then, a linearization is obtained at $\mathbf{w}_{t+1}^c$. This cut iterate adds variability to the algorithm to prevent it from stalling at $\mathbf{w}_t^b$ when $\eta = 0$. Franc and Sonnenburg [2008] reported that setting $\theta = 0.9$ works well in practice. The algorithm terminates when $J(\mathbf{w}_{t+1}^b) - J_t(\mathbf{w}_{t+1}) \leq \epsilon$ for some pre-defined accuracy $\epsilon > 0$. Algorithm 3 summarizes the procedure.

The advantage of this variant is that the iterates $\mathbf{w}_i^b$ ensure non-increasing sequence of objective values $J(\mathbf{w}_i^b)$, whereas the sequence $J(\mathbf{w}_i)$ may fluctuate in BMRM. In other words, the line search step stabilizes the BMRM algorithm.

Although the line search step provides stabilization to the original BMRM algorithm and reduces the number of iterations to convergence, it increases the per iteration runtime complexity. Depending on the type of line search used, *e.g.*, exact or inexact [Nocedal and Wright, 1999], the runtime can increase dramatically. Fortunately, for certain regularized risks such as linear SVMs and multiclass SVMs [Crammer and Singer, 2003], an exact line search can be done efficiently. See Franc and Sonnenburg [2008] for an exact line search for binary hinge loss and Yu et al. [2008] for detailed discussions on binary and multiclass line searches.

Algorithm 3 BMRM with Line Search

input: $\epsilon > 0$, $\theta \in (0, 1]$, $\mathbf{w}_1^b \in \mathbb{R}^d$
initialization: $\mathbf{w}_1^c \leftarrow \mathbf{w}_1^b$, $t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  Compute $\mathbf{a}_t \in \partial R(\mathbf{w}_t^c)$ and $b_t \leftarrow R(\mathbf{w}_t^c) - \langle \mathbf{w}_t^c, \mathbf{a}_t \rangle$
  Update $\check{R}_t(\mathbf{w}) := \max_{i \in [t]} \{\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i\}$
  $\mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w}} J_t(\mathbf{w}) := \lambda \Omega(\mathbf{w}) + \check{R}_t(\mathbf{w})$
  [Linesearch] $\eta_t \leftarrow \operatorname{argmin}_{\eta \in \mathbb{R}} J(\mathbf{w}_t^b + \eta(\mathbf{w}_{t+1} - \mathbf{w}_t^b))$
  $\mathbf{w}_{t+1}^b \leftarrow \mathbf{w}_t^b + \eta_t(\mathbf{w}_{t+1} - \mathbf{w}_t^b)$
  $\mathbf{w}_{t+1}^c \leftarrow (1 - \theta)\, \mathbf{w}_{t+1}^b + \theta\, \mathbf{w}_{t+1}$
  $\epsilon_t \leftarrow J(\mathbf{w}_{t+1}^b) - J_t(\mathbf{w}_{t+1})$
until $\epsilon_t \leq \epsilon$
return $\mathbf{w}_{t+1}^b$

3.2 Convergence Analysis

While the variants of bundle methods we proposed are intuitively plausible, it remains to be shown that they have good rates of convergence. In fact, past results, such as those by Tsochantaridis et al. [2005] suggest a slow $O(\epsilon^{-2})$ rate of convergence. In this section we tighten their results and show an $O(\epsilon^{-1})$ rate of convergence for non-differentiable risks and $O(\log(\epsilon^{-1}))$ rates for differentiable risks under some mild assumptions. More concretely we prove the following two convergence results:

(a) Assume that the empirical risk R is at least locally Lipschitz continuous on $\mathbb{R}^d$. For σ-strongly convex regularizer Ω we prove that Algorithm 2 converges to within ϵ-accurate solution in $O((\lambda\sigma\epsilon)^{-1})$ iterations.

(b) Under the above conditions, assume further that J (is differentiable and) has H-Lipschitz continuous gradient (see Definition C.0.8) *i.e.*,

$$\|\nabla J(\mathbf{u}) - \nabla J(\mathbf{v})\| \leq H \|\mathbf{u} - \mathbf{v}\|, \forall\ (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^d \times \mathbb{R}^d.$$

In other words, the second order derivative of J is bounded, if it exists. Under these conditions, we prove a stronger $O((\lambda\sigma)^{-1} \log(\epsilon^{-1}))$ rate of convergence.

For our convergence proofs we use a duality argument similar to those put forward in Shalev-Shwartz and Singer [2006] and Tsochantaridis et al. [2005], both of which share key techniques with Zhang [2003]. Recall that ϵ_t denotes our approximation gap, which in turn upper bounds how far away we are from the optimal solution. In other words, $\epsilon_t \geq \min_{i \in [t]} J(\mathbf{w}_i) - J^*$, where J^* denotes the optimum value of the objective function J. The quantity $\epsilon_t - \epsilon_{t+1}$ can thus be viewed as the "progress" made towards J^* in iteration t. The crux of our proof argument lies in showing that for non-differentiable empirical risks the recurrence $\epsilon_t - \epsilon_{t+1} \geq c \cdot \epsilon_t^2$ holds for some appropriately chosen

constant c. The rates then follow by invoking a lemma from Abe et al. [2001]. In the case of the differentiable empirical risks we show that $\epsilon_t - \epsilon_{t+1} \geq c' \cdot \epsilon_t$ thus implying an $O(\log(\epsilon^{-1}))$ rate of convergence.

In order to show the required recurrence, we first observe that by strong duality the values of the primal and dual problems (3.3) and (3.4) are equal at optimality. Hence, any progress in J_{t+1} can be computed in the dual. Next, we observe that the solution of the dual problem (3.4) at iteration t, denoted by $\boldsymbol{\alpha}_t$, forms a feasible set of parameters for the dual problem (3.4) at iteration $t+1$ by means of the parameterization $(\boldsymbol{\alpha}_t, 0)$, *i.e.*, by padding $\boldsymbol{\alpha}_t$ with a 0.

To obtain a lower bound on the improvement of the approximation *i.e.*, $\min_{\mathbf{w}} J_{t+1}(\mathbf{w}) - \min_{\mathbf{w}} J_t(\mathbf{w})$, due to the addition of new linearization at $\mathbf{w}_{t+1}$ we perform a 1-dimensional optimization along $((1-\eta)\boldsymbol{\alpha}_t, \eta)$ in (3.4). The constraint $\eta \in (0,1)$ ensures dual feasibility. We will then bound this improvement in terms of ϵ_t. Note that, in general, solving the dual problem (3.4) results in an improvement which is larger than that obtained via the 1-dimensional optimization which is used only for analytic tractability. We now state our key theorem and prove it in Appendix B.3.

Theorem 3.2.1. *Assume that* $\sup_i \|\mathbf{a}_i\| \leq G$ *where* $\mathbf{a}_i \in \partial R(\mathbf{w}_i)$ *and* $\mathbf{w}_i$ *are subgradients and iterates generated by Algorithm 2. Also assume that* Ω *is* σ*-strongly convex. In this case we have*

$$\epsilon_t - \epsilon_{t+1} \geq \frac{\epsilon_t}{2} \min(1, \lambda\sigma\epsilon_t/(4G^2)). \tag{3.6}$$

Furthermore, if J *has* H*-Lipschitz continuous gradient, then we have*

$$\epsilon_t - \epsilon_{t+1} \geq \begin{cases} \epsilon_t/2 & \text{if } \epsilon_t > 4G^2/(\lambda\sigma) \\ \lambda\sigma/8 & \text{if } 4G^2/(\lambda\sigma) \geq \epsilon_t \geq H/2 \\ \lambda\sigma\epsilon_t/(4H) & \text{otherwise} \end{cases}$$

Note that the error keeps on halving initially and settles for a somewhat slower rate of convergence after that, whenever the second order derivative of J is bounded from above. The reason for the difference in the convergence bound for differentiable and non-differentiable empirical risks is that in the former case the gradient of the risk converges to 0 as we approach optimality, whereas in the latter case, no such guarantees hold.[1] We are now in a position to state our main convergence result. The proof can be found in Appendix B.4.

Theorem 3.2.2. *Under the assumptions of Theorem 3.2.1 we can give the following convergence guarantee for Algorithm 2. For any* $\epsilon < 4G^2/(\lambda\sigma)$ *the algorithm converges*

[1] For example, when minimizing $\|\mathbf{w}\|_1$ the (sub)gradient may not vanish at the optimum $\mathbf{w}^* = \mathbf{0}$.

to the desired precision after

$$T \leq \log_2 \frac{\lambda\sigma\epsilon_1}{4G^2} + \frac{8G^2}{\lambda\sigma\epsilon} - 1$$

steps. Furthermore if J has H-Lipschitz continuous gradient, convergence to any $\epsilon \leq H/2$ takes at most the following number of steps:

$$T \leq \log_2 \frac{\lambda\sigma\epsilon_1}{4G^2} + \frac{4}{\lambda\sigma} \max\left(0, H - 8G^2(\lambda\sigma)^{-1}\right) + \frac{4H}{\lambda\sigma} \log \frac{H}{2\epsilon}$$

Several observations are in order: First, note that the number of iterations only depends *logarithmically* on the first approximation gap ϵ_1 which essentially upper bounds the difference between the initial value $J(\mathbf{w}_1)$ and the optimal solution $\min_{\mathbf{w}} J(\mathbf{w})$. Therefore, a badly chosen initial point will not affect the overall performance of Algorithm 2 much. In the cases where a lower bound $\underline{J}$ of $J(\mathbf{w})$ is known in advance[2] ϵ_1 can be replaced by the constant $J(\mathbf{w}_1) - \underline{J}$.

Second, the rate has an $O(\epsilon^{-1})$ dependence in the non-differentiable case, as opposed to the $O(\epsilon^{-2})$ rates of Tsochantaridis et al. [2005], and the $O(\epsilon^{-1}\log(\epsilon^{-1}))$ rate of the proximal bundle method (*cf.* Theorem 2.3.1). In addition to that, the convergence rate is $O(\log(\epsilon^{-1}))$ for continuously differentiable problems which matches the convergence rate of gradient descent method with line search [see Boyd and Vandenberghe, 2004, Section 9.3.1].

For completeness we also state the convergence guarantees for Algorithm 3 and provide a proof in Appendix B.5.

Theorem 3.2.3. *Under the assumptions of Theorem 3.2.1 Algorithm 3 converges to the desired precision ϵ after*

$$T \leq \frac{8G^2}{\lambda\sigma\epsilon} - 1$$

steps for any $\epsilon < 4G^2/(\lambda\sigma)$.

3.3 Implementation

There are many machine learning problems such as document classification where the feature dimension d and the number of training examples m are large. Furthermore, the training environments are usually readily equipped with modern computing resources such as computer clusters or computers with multiple cores (*i.e.*, processing units). We

[2] Combination of regularizers in Section 3.1.1 and max-margin/logistic empirical risks have a lower bound of 0.

discuss here various software design and implementation issues of BMRM targeting the scenario where d and m are large, and/or computer clusters are available for use.

3.3.1 Modularity and Generality

From Algorithm 2, we see two distinct computationally intensive stages: generating linearizations (*i.e.*, subgradient $\mathbf{a}_i$ and offset b_i) which requires computing the value and derivative of empirical risk R, and subsequently solving a concave (or convex) optimization problem. Despite the sequential nature of the computation, the algorithm admits a modular implementation.

Figure 3.1 depicts the software architecture of our implementation of BMRM. There are fundamentally three decoupled modules namely, "Optimizer", "Loss", and "Data", which constitute the software implementation:

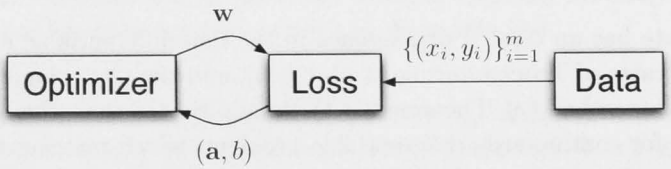


Figure 3.1: Software architecture of BMRM.

- The role of the Optimizer module is to generate new iterate $\mathbf{w}_{t+1}$ by solving (3.1) upon receiving a new linearization from the Loss module at each iteration. After that, the new iterate is passed back to the Loss module to start the next iteration. Since the subproblem (3.1) is closely tied to the choice of regularizer, the implementation of Optimizer module is in general different for different regularizers. For instance, the module implements a quadratic program solver for squared L_2 norm and quadratic form regularizers, and a constrained quasi-Newton algorithm for negative entropy regularizers.

- The Loss module implements the loss function of choice. In particular, at each iteration, the Loss module first receives an iterate $\mathbf{w}$ from the Optimizer module and training examples (x_i, y_i) from the Data module. Then it computes the subgradient $\mathbf{a}$ and offset b that will be passed to Optimizer module for the computation of new iterate.

- The Data module manages training examples. Its main functionalities include retrieving data from local or remote disks, and, if necessary, converting training examples into appropriate form so that the Loss module can evaluate loss on the examples.

The modular design promotes code reuse hence reduces the development time. Moreover, the generality of bundle methods is fully reflected in this design as the risk/loss and regularizer implementations are independent of each other.

3.3.2 Scalability via Parallel and Distributed Computation

Note that the empirical risk is usually defined as a weighted sum of loss functions over the training examples which are independent of each other. Hence, the computation of empirical risk can be parallelized and distributed over multiple machines. The parallelization or distribution is done in a way such that each participating machine holds a non-overlapping subset of the whole set of training examples.

Figure 3.2 illustrates the parallelization and distribution of empirical risk computation: In each iteration, the Optimizer module broadcasts current iterate $\mathbf{w}$ to all participating machines through the (conceptual) Multiplexer module. These machines in turn compute the values and subgradients of the loss function on their subsets of examples. These values and subgradients are then propagated back to the Optimizer module via the Multiplexer module which aggregates the subgradients and loss values into its final form of linearization. We note that the broadcasts of iterate and aggregation of subgradients can be done efficiently in time logarithmic in the number of machines using standard implementations of parallel and distributed computation protocol such as MPICH2 [Gropp et al., 1999].

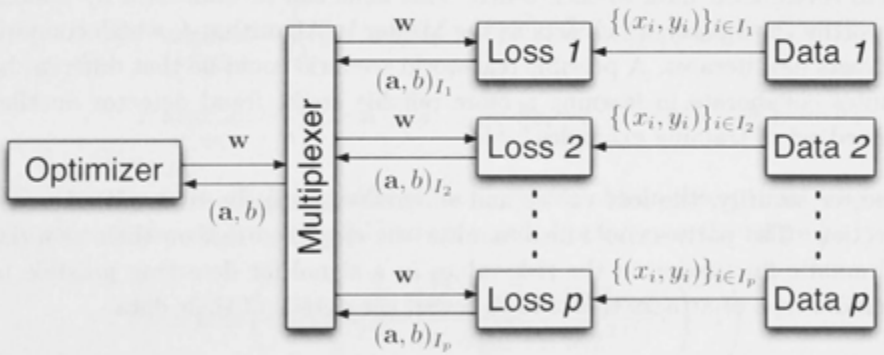


Figure 3.2: Software architecture of BMRM in parallel/distributed computation setting.

There are two immediate benefits of parallelizing and distributing the computation: First, parallelization of empirical risk computation over multiple machines/cores leads to significant speedup when the loss function evaluation is computationally intensive. Second, training examples are distributed to multiple machines hence a large set of training examples which cannot be loaded fully into the memory of a single machine can be done so on multiple machines, albeit distributed.

For completeness, we provide here a version of Algorithm 2 adapted to the master-slave parallel computation model in Algorithm 4.

Algorithm 4 Parallel BMRM

input: $\epsilon > 0$, $\mathbf{w}_1 \in \mathbb{R}^d$, dataset S, number of slave machines p
initialization: $t \leftarrow 0$, assign sub-dataset S_i to slave i, $i = 1, \ldots, p$
repeat
  $t \leftarrow t + 1$
  Master: Broadcast $\mathbf{w}_t$ to all slaves
  Slaves: Compute $R^i(\mathbf{w}_t) := \sum_{(x,y)\in S_i} l(x, y, w_t)$ and $\mathbf{a}_t^i \in \partial R^i(\mathbf{w}_t)$
  Master: Aggregate $\mathbf{a}_t := \frac{1}{|S|} \sum_{i=1}^p \mathbf{a}_t^i$ and $b_t := \frac{1}{|S|} \sum_{i=1}^p R^i(\mathbf{w}_t) - \langle \mathbf{w}_t, \mathbf{a}_t \rangle$
  Master: Update $\check{R}_t(\mathbf{w}) := \max_{i\in[t]} \{\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i\}$
  Master: $\mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w}} J_t(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + \check{R}_t(\mathbf{w})$
  Master: $\mathbf{w}_* := \operatorname{argmin}_{k\in[t+1]} J(\mathbf{w}_k)$
  Master: $\epsilon_t \leftarrow J(\mathbf{w}_*) - J_t(\mathbf{w}_t)$
until $\epsilon_t \leq \epsilon$
return $\mathbf{w}_*$

3.3.3 Privacy Preserving Collaborative Learning

One additional application of parallel and distributed computation of empirical risk is the privacy preserving collaborative learning. In this setting, a few possibly geographically separated parties are willing to contribute their data for learning but do not want to reveal their data to each other. This issue can be addressed by appointing a trustworthy third party which acts as the Master in Algorithm 4, which computes and broadcasts new iterates. A possible real-world scenario could be that different banking institutes collaborate in learning a more reliable credit fraud detector on the *larger* combined set of training examples.

For better security, the loss values and subgradients can be transmitted over secure connection. The parties could also monitor the empirical risk on their own data and use dramatic fluctuation in the risk values as a signal for detecting possible man-in-the-middle type of attacks that aim to reveal the details of their data.

3.3.4 Memory Space Efficiency

The convergence proof of BMRM relies on the past linearizations collectively, that is, there is no difference if the linearizations are stored separately or aggregated in a way very similar to the linearization aggregation in the proximal bundle method. It is also true that BMRM can perform linearization selection *i.e.*, dropping linearizations with zero Lagrange multipliers as in the proximal bundle method. We present in this section the linearization aggregation technique for BMRM.

Note that at iteration t, before the computation for new iterate $\mathbf{w}_{t+1}$, Algorithm 2 maintains a bundle of t subgradients $\{\mathbf{a}_i\}_{i=1}^{t}$ of R computed at the locations $\{\mathbf{w}_i\}_{i=1}^{t}$. Furthermore, the Lagrange multipliers $\boldsymbol{\alpha}_{t-1}$ obtained in iteration $t-1$ satisfy $\boldsymbol{\alpha}_{t-1} \geq \mathbf{0}_{t-1}$ and $\|\boldsymbol{\alpha}\|_1 = 1$ by the constraints of (3.4). We define the *aggregated* (sub)gradient $\tilde{\mathbf{a}}^I$, offset $\tilde{b}^I$ and Lagrange multiplier $\tilde{\alpha}_{t-1}^I$ as

$$\tilde{\mathbf{a}}^I := \frac{1}{\tilde{\alpha}_{t-1}^I} \sum_{i \in I} \alpha_{t-1,i}\, \mathbf{a}_i, \quad \tilde{b}^I := \frac{1}{\tilde{\alpha}_{t-1}^I} \sum_{i \in I} \alpha_{t-1,i} b_i, \quad \text{and} \quad \tilde{\alpha}_{t-1}^I := \sum_{i \in I} \alpha_{t-1,i},$$

respectively, where $I \subseteq [t-1]$ is an index set [Kiwiel, 1983]. Clearly, the optimality of (3.4) at the end of iteration $t-1$ is maintained when a subset $\{(\mathbf{a}_i, b_i, \alpha_{t-1,i})\}_{i \in I}$ is replaced by the aggregate $(\tilde{\mathbf{a}}^I, \tilde{b}^I, \tilde{\alpha}_{t-1}^I)$ for any $I \subseteq [t-1]$.

To obtain a new iterate $\mathbf{w}_{t+1}$ via (3.4) with memory space for at most k linearizations, we can, for example, replace $\{(\mathbf{a}_i, b_i)\}_{i \in I}$ with $(\tilde{\mathbf{a}}^I, \tilde{b}^I)$ where $I = [t-k+1]$ and $2 \leq k \leq t$. Then, we solve a k-dimensional variant of (3.4) with $\mathbf{A}_t := [\tilde{\mathbf{a}}^I, \mathbf{a}_{t-k+2}, \ldots, \mathbf{a}_t]$, $\mathbf{b}_t := [\tilde{b}^I, b_{t-k+2}, \ldots, b_t]$, and $\boldsymbol{\alpha} \in \mathbb{R}^k$. The optimum of this variant will be lower than or equal to that of (3.4) as the latter has higher degree of freedom than the former. Nevertheless, solving this variant with $2 \leq k \leq t$ will still guarantee convergence (recall that our convergence proof only uses $k = 2$). In the sequel we name the aforementioned number k as the "bundle size" since it indicates the number of linearizations the algorithm keeps.

To illustrate, we provide here a memory efficient BMRM variant for the case where $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ and $k = 2$. We first see that the dual of subproblem (3.1) now reads:

$$\begin{aligned} \eta &= \operatorname*{argmax}_{0 \leq \eta \leq 1} -\frac{1}{2\lambda} \left\| \tilde{\mathbf{a}}^I + \eta(\mathbf{a}_t - \tilde{\mathbf{a}}^I) \right\|_2^2 + \tilde{b}^I + \eta(b_t - \tilde{b}^I) \\ &\equiv \operatorname*{argmax}_{0 \leq \eta \leq 1} -\frac{\eta}{\lambda}(\mathbf{a}_t - \tilde{\mathbf{a}}^I)^\top \tilde{\mathbf{a}}^I - \frac{\eta^2}{2\lambda} \left\| \mathbf{a}_t - \tilde{\mathbf{a}}^I \right\|^2 + \eta(b_t - \tilde{b}^I) \end{aligned} \tag{3.7}$$

where $I := [t-1]$. Since (3.7) is quadratic in η, we can obtain the optimal η by setting the derivative of the objective in (3.7) to zero and clipping η in the range $[0, 1]$:

$$\eta = \min\left(\max\left(0, \frac{b_t - \tilde{b}^I + \mathbf{w}_t^\top \mathbf{a}_t + \lambda \|\mathbf{w}_t\|^2}{\lambda^{-1} \|\mathbf{a}_t + \lambda \mathbf{w}_t\|^2} \right), 1 \right) \tag{3.8}$$

where $\mathbf{w}_t = -\frac{1}{\lambda}\tilde{\mathbf{a}}^I$ by the dual connection. With the optimal η, we obtain the new primal iterate $\mathbf{w}_{t+1} = (1-\eta)\,\mathbf{w}_t - (\eta/\lambda)\,\mathbf{a}_t$. Algorithm 5 lists the details. Note that this variant is simple to implement and does not require a QP solver.

Algorithm 5 Memory efficient BMRM

input: $\epsilon > 0$, $\mathbf{w}_1 \in \mathbb{R}^d$
initialization: $t \leftarrow 1$
Compute $\mathbf{a}_1 \in \partial R(\mathbf{w}_1)$, and $b_1 \leftarrow R(\mathbf{w}_1) - \langle \mathbf{w}_1, \mathbf{a}_1 \rangle$
$\mathbf{w}_2 \leftarrow -\frac{1}{\lambda} \mathbf{a}_1$
$\tilde{b}^I \leftarrow b_1$
repeat
  $t \leftarrow t + 1$
  Compute $\mathbf{a}_t \in \partial R(\mathbf{w}_t)$ and $b_t \leftarrow R(\mathbf{w}_t) - \langle \mathbf{w}_t, \mathbf{a}_t \rangle$
  Compute η using Eq. (3.8)
  $\mathbf{w}_{t+1} \leftarrow (1 - \eta)\, \mathbf{w}_t - (\eta/\lambda)\, \mathbf{a}_t$
  $\tilde{b}^I \leftarrow (1 - \eta)\tilde{b}^I + \eta b_t$
  $* := \operatorname{argmin}_{k \in [t+1]} J(\mathbf{w}_k)$
  $\epsilon_t \leftarrow J(\mathbf{w}_*) - J_t(\mathbf{w}_{t+1})$
until $\epsilon_t \leq \epsilon$
return $\mathbf{w}_*$

3.4 Experiments

In this section, we examine the convergence behavior of BMRM and show that it is versatile enough to solve a variety of machine learning problems *i.e.*, with differentiable and non-differentiable empirical risks. All our experiments were carried out on a cluster of 16 machines running Rocks Cluster Distribution version 4.3. Each machine has a 2.4GHz AMD Dual Core processor and 4GB of RAM. Details of the loss functions, datasets, competing solvers and experimental objectives are described in the following sections.

3.4.1 Convergence Behavior

We investigated the convergence rate of Algorithm 2 empirically with respect to regularization parameter λ, approximation gap ϵ, and bundle size k. In addition, we investigated the speedup gained by parallelizing the empirical risk computation. Finally, we examined empirically how generalization performance is related to approximation gap. We focused on the training of binary classifier with linear SVM:

$$\min_{\mathbf{w}} J(\mathbf{w}) := \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \langle \mathbf{w}, x_i \rangle), \tag{3.9}$$

where $y_i \in \{\pm 1\}$ and $x_i \in \mathbb{R}^d$.

The experiments were conducted on 6 datasets commonly used in binary classification

studies, namely, adult9, astro-ph, news20-b[3], rcv1, real-sim, and worm. adult9, news20-b, rcv1[4], and real-sim are available on the LIBSVM tools website[5]. astro-ph [Joachims, 2006] and worm [Franc and Sonnenburg, 2008] are available upon request from Thorsten Joachims and Soeren Sonnenburg, respectively. Table 3.1 summarizes the properties of the datasets.

Dataset	#examples m	dimension d	density (%)
adult9	48,842	123	11.27
astro-ph	94,856	99,757	0.08
news20-b	19,954	1,355,191	0.03
rcv1	677,399	47,236	0.15
real-sim	72,201	20,958	0.25
worm	1,026,036	804	25.00

Table 3.1: Properties of the binary classification datasets used in the experiments. The density of a dataset is computed as the total number of non-zero features multiplied by $100/(md)$.

Regularization Parameter λ and Approximation Gap ϵ

As suggested by the convergence analysis, the linear SVM with the non-differentiable binary hinge loss should converge in $O((\lambda\epsilon)^{-1})$ iterations, where λ and ϵ are two parameters which one normally tunes during the model selection phase. Therefore, we investigated the scaling behavior of Algorithm 2 w.r.t. these two parameters. We performed the experiments with no limit on the bundle size but with a heuristic that removes linearization which remained inactive (*i.e.*, Lagrange multiplier being zero) for 10 or more consecutive iterations.[6]

Figure 3.3 shows the approximation gap ϵ_t as a function of number of iterations t. As predicted by the convergence analysis, BMRM converges faster for larger values of λ. Furthermore, the empirical convergence curves exhibit a $O(\log(\epsilon^{-1}))$ rate instead of the (pessimistic) theoretical rate of $O(\epsilon^{-1})$, especially for large values of λ. Interestingly, BMRM converges faster on high-dimensional text datasets (*i.e.*, astro-ph, news20-b, rcv1, and real-sim) than on lower dimensional datasets (*i.e.*, adult9 and worm).

[3]The dataset is originally named news20; we renamed it to avoid confusion with the multiclass version of the dataset.

[4]The test set of rcv1 was used for training instead of the smaller training set which only contains about 23K examples.

[5]`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

[6]This heuristic does not have any implication on the convergence analysis.

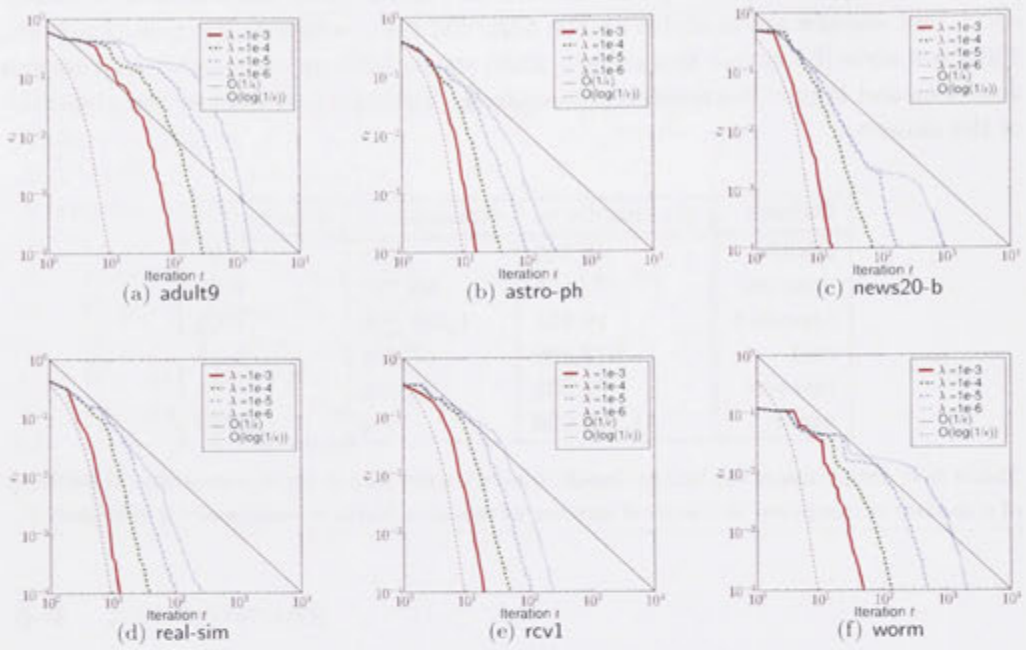


Figure 3.3: Approximation gap ϵ_t as a function of number of iterations t; for different regularization parameters λ (and unlimited bundle size).

Bundle Size

The dual subproblem (3.4) is a concave problem which has dimensionality equal to the number of iterations executed. In the case of linear SVM, (3.4) is exactly a QP problem. Hence, as described in Section 3.3.4, we can trade potentially greater improvement per iteration for memory efficiency.

Figure 3.4 shows the approximation gap ϵ_t during the training of linear SVM as a function of the number of iterations t, for different bundle sizes $k \in \{2, 10, 50, \infty\}$. In the case of $k = \infty$, we did not limit the bundle size and employed the same heuristics which removes inactive linearizations as mentioned in section 3.4.1. As expected, the algorithm converged faster for larger k. Although the case $k = 2$ was the slowest, its convergence rate was still faster than the theoretical bound $O((\lambda\epsilon)^{-1})$.

Parallelization

We performed experiments for linear SVMs training with parallelized risk computation on the worm dataset. Figure 3.5(a) shows the wallclock time for the overall training phase (*e.g.*, data loading, risk computation, and solving the QP) and CPU time for

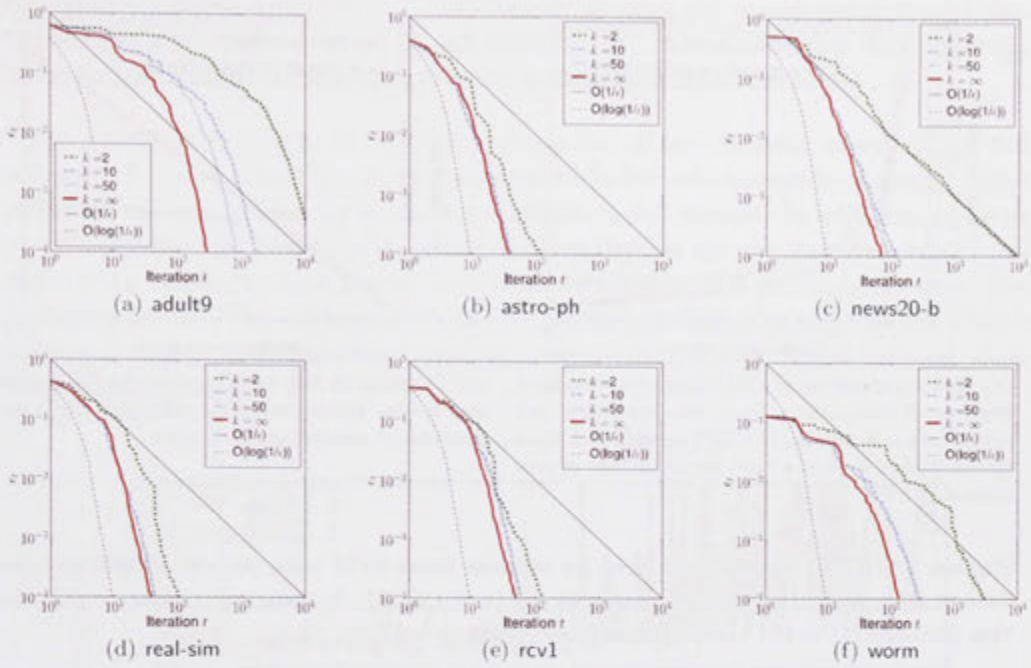


Figure 3.4: Approximation gap ϵ_t as a function of number of iterations t; for different bundle sizes k (and fixed regularization parameter $\lambda = 10^{-4}$).

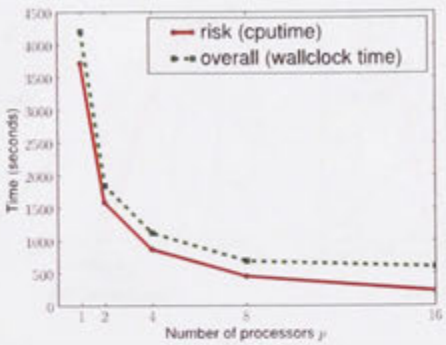
just the risk computation as a function of number of processors p. Note that the gap between the two curves essentially tells the runtime upper bound of the sequential part of the algorithm. We see that both overall and risk computation time decreased as the number of processors p was increased. However, in Figure 3.5(b), we see two different speedups.[7] The speedup for the risk computation is roughly linear as there is no sequential part in it; the speedup of overall computation is approaching a limit[8] which is well-explained by Amdahl's law [Amdahl, 1967].

Generalization Versus Approximation Gap

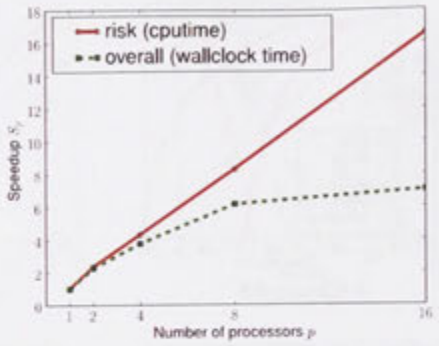
Since the problems we are considering are convex, all properly convergent optimizers will converge to the same solution. Therefore, comparing generalization performance of the final solution is meaningless. But, in real life one is often interested in the speed with which the algorithm achieves good generalization performance. In this section we study this question. We focus on the generalization (in terms of accuracy) as a function of approximation gap during training. For this experiment, we randomly split each of

[7] Speedup $S_p = \frac{T_1}{T_p}$ where p is the number of processors and T_q is the runtime of the parallelized algorithm on q processors.

[8] The limit of speedup is the inverse of the sequential fraction of the algorithm such as the QP.



(a) Risk computation in CPU time (red solid line) and overall computation (*i.e.*, data loading + risk computation + solving the QP) in wallclock time (green dashed line) as a function of number of processors.



(b) Speedup in risk computation (in CPU time) and overall computation (in wallclock time) as a function of number of processors.

Figure 3.5: CPU and wallclock time for training linear SVM using parallel BMRM on worm dataset with varying number of processors $p \in \{1, 2, 4, 8, 16\}$. In these experiments, regularization parameter $\lambda = 10^{-6}$, and termination criterion $\epsilon = 10^{-4}$.

the datasets into training (60%), validation (20%) and testing (20%) sets.

We first obtained the best $\lambda \in \{2^{-20}, \ldots, 2^{0}\}$ for each of the datasets using their corresponding validation sets. With these best λ's, we (re)trained linear SVMs and recorded the testing accuracy as well as the approximation gap at every iteration, with termination criterion $\epsilon = 10^{-4}$. Figure 3.6 shows the difference between the testing accuracy evaluated at every iteration and that after training, as a function of approximation gap at each iteration.

From the figure, we see that the testing accuracies for adult9 and worm datasets are less stable in general and the approximation gap must be reduced to at least 10^{-3} to reach the 0.5% regime of the final testing accuracies; the testing accuracies for the rest of the datasets arrived at the same regime with approximation gap of 10^{-2} or lower.

In general, the generalization improved as the approximation gap is decreased. The improvement in generalization became rather insignificant (say, the maximum of changes in testing accuracies is less than 0.1%) when the approximation gap was further reduced to below some effective threshold ϵ_{eff}; that said, it is not necessary to continue the optimization when $\epsilon_t \leq \epsilon_{\text{eff}}$.[9] Since ϵ_{eff} (or its scale) is not known *a priori* and the asymptotic analysis in Shalev-Schwartz and Srebro [2008] does not reveal the actual scale of ϵ_{eff} directly applicable in our case, we carried out another set of experiments to investigate if ϵ_{eff} could be estimated with as little effort as possible: For each dataset,

[9] Heuristically, we could terminate the training phase following the *early stopping* strategy by monitoring the changes in accuracies on validation set evaluated in some most recent iterations.

we randomly subsampled $10\%, \ldots, 50\%$ of the training set as sub-datasets and performed the same experiment on all sub-datasets. We then determined the largest ϵ_{eff} such that the maximum changes in testing accuracies is less than 0.1%.

Table 3.2 shows the (base 10 logarithm of) ϵ_{eff} for all sub-datasets as well as the full datasets. It seems that the ϵ_{eff} estimated on a smaller sub-dataset is at most 1 order of magnitude larger than the actual ϵ_{eff} required on full dataset. In addition, we show in the table that the necessary threshold $\epsilon_{10\%}$ required by the sub-datasets and the full datasets to attain the final testing accuracies attained by the 10% sub-datasets. The observations obey the analysis in Shalev-Schwartz and Srebro [2008] that for a fixed testing accuracy, approximation gap (*i.e.*, optimization error) can be relaxed when more data is given.

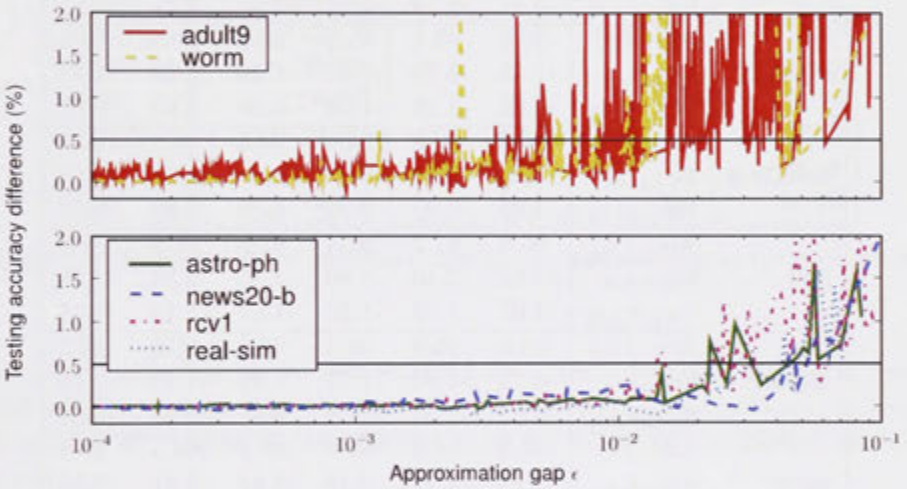


Figure 3.6: Difference between testing accuracies of intermediate and final models.

3.4.2 Comparison with Standard Bundle Methods

In this section we compared BMRM with BT, an implementation of the Trust-Region bundle method (see Section 2.3.4) obtained from Schramm and Zowe [1992], and with the variant of BMRM with line search, *i.e.*, LSBMRM (Algorithm 3). Details of the multiclass line search used in LSBMRM can be found in Yu et al. [2008].

For binary classification, we solve the linear SVM (3.9) on the datasets: adult9, astro-ph, news20-b, rcv1, real-sim, and worm as mentioned in Section 3.4.1. For multiclass

		Proportion of training set used					
		10%	20%	30%	40%	50%	100%
adult9	Acc. (%)	84.3	84.7	84.9	85.1	85.1	85.2
	$\log_{10}\epsilon_{\text{eff}}$	-3.90	-3.72	-3.77	-3.88	-3.64	-4.00
	$\log_{10}\epsilon_{10\%}$	-4.01	-1.18	-1.07	-1.16	-1.27	-1.04
astro-ph	Acc. (%)	96.1	96.6	96.4	96.6	96.8	97.4
	$\log_{10}\epsilon_{\text{eff}}$	-1.48	-1.70	-1.57	-1.49	-1.68	-1.84
	$\log_{10}\epsilon_{10\%}$	-4.00	-1.15	-1.06	-0.98	-1.02	-0.87
news20-b	Acc. (%)	89.9	92.9	94.3	94.5	95.4	96.6
	$\log_{10}\epsilon_{\text{eff}}$	-2.00	-2.48	-3.87	-1.65	-3.71	-2.84
	$\log_{10}\epsilon_{10\%}$	-4.02	-0.92	-0.70	-0.80	-0.80	-0.67
rcv1	Acc. (%)	96.9	97.2	97.4	97.2	97.5	97.6
	$\log_{10}\epsilon_{\text{eff}}$	-2.02	-2.40	-1.99	-2.16	-2.34	-2.28
	$\log_{10}\epsilon_{10\%}$	-4.07	-1.19	-1.30	-1.29	-1.13	-1.11
real-sim	Acc. (%)	95.0	95.9	96.3	96.6	96.6	97.2
	$\log_{10}\epsilon_{\text{eff}}$	-1.74	-1.84	-1.71	-1.99	-1.74	-1.75
	$\log_{10}\epsilon_{10\%}$	-4.02	-1.04	-0.88	-0.87	-0.85	-0.82
worm	Acc. (%)	98.2	98.2	98.2	98.3	98.3	98.4
	$\log_{10}\epsilon_{\text{eff}}$	-2.43	-2.47	-2.48	-3.62	-2.81	-3.55
	$\log_{10}\epsilon_{10\%}$	-4.00	-1.38	-1.28	-1.37	-1.28	-1.31

Table 3.2: The first sub-row in each dataset row indicates the testing accuracies of models trained on the corresponding proportions of the training set. The second sub-row indicates the (base 10 logarithm of) effective threshold such that the maximum difference in testing accuracies of models with approximation gap smaller than that is less than 0.1%. The third sub-row indicates the (base 10 logarithm of) threshold necessary for models to attain the testing accuracy attained by the model trained on the 10% sub-dataset with default $\epsilon = 10^{-4}$.

classification, we solve [Crammer and Singer, 2003]:

$$\min_{\mathbf{w}} J(\mathbf{w}) := \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max_{y_i' \in [c]} \left\langle \mathbf{w}, e_{y_i'} \otimes x_i - e_{y_i} \otimes x_i \right\rangle + \mathbf{I}(y_i \neq y_i'), \tag{3.10}$$

where c is the number of classes in the problem, e_i is the i-th standard basis for $\mathbb{R}^c$, $\otimes$ denotes Kronecker product; and $\mathbf{I}(\cdot)$ is an indicator function that has value 1 if its argument is evaluated true, and 0 otherwise. The datasets used in multiclass classification experiments were inex, letter, mnist, news20-m[10], protein, and usps. inex is available for download on the website of Antoine Bordes[11] and the rest can be found on the LIBSVM tools website[12]. Table 3.3 summarizes the properties of the multiclass datasets.

Dataset	#examples m	#classes c	dimension d	density %
inex	12,107	18	167,295	0.48
letter	20,000	26	16	100.00
mnist	70,000	10	780	19.24
news20-m	19,928	20	62,061	0.13
protein	21,516	3	357	28.31
usps	9,298	10	256	96.70

Table 3.3: Properties of the multiclass classification datasets used in the experiments.

In each of the experiments, we first obtained the optimal weight vector $\mathbf{w}_*$ by running BMRM until the termination criterion $J(\mathbf{w}_{t+1}) - J_t(\mathbf{w}_{t+1}) \leq 0.01 J(\mathbf{w}_{t+1})$ is satisfied. Then we run BT, LSBMRM, and BMRM until the following termination criterion is satisfied:

$$J(\mathbf{w}_{t+1}) - J(\mathbf{w}_*) \leq 0.01 J(\mathbf{w}_*). \tag{3.11}$$

Figure 3.7 shows the number of iterations t required by the three methods on each dataset to satisfy (3.11) as a function of regularization parameter $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. As expected, LSBMRM, which uses an exact line search, outperformed both BMRM and BT on all datasets. BMRM performed better than BT on all high dimensional datasets except news20-m but worse on the rest. Although BT tunes the stabilization trade-off parameter τ_t automatically, it still does not guarantee superiority over BMRM which is considerably simpler. Nevertheless, external stabilization (in BT) clearly helps speed up the convergence in certain cases, especially when λ is small.

[10] The dataset is originally named news20; we renamed it to avoid confusion with the binary version of the dataset.

[11] http://webia.lip6.fr/~bordes/datasets/multiclass/inex.tar.gz

[12] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

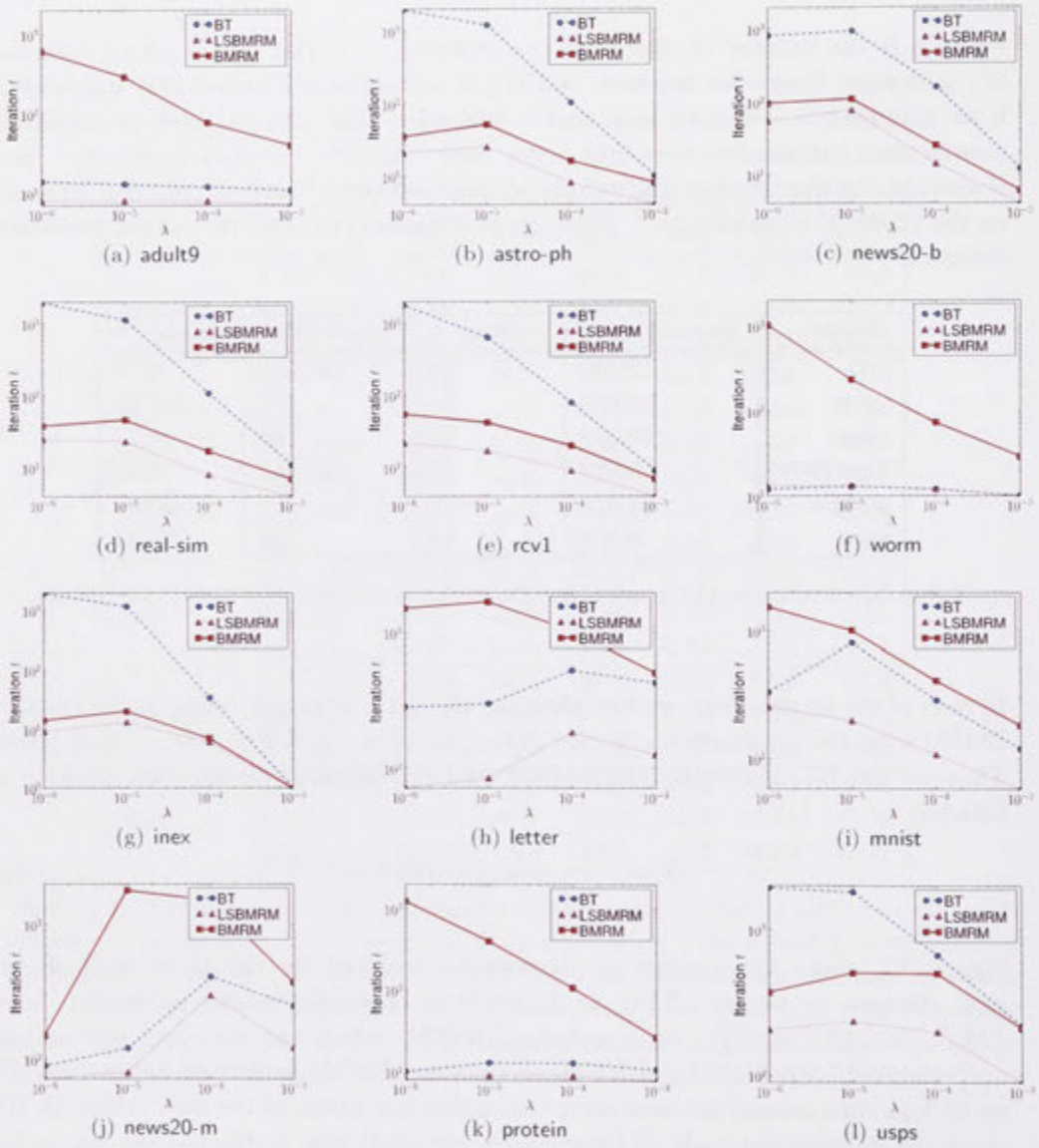


Figure 3.7: Smallest number of iterations required to satisfy the termination criterion (3.11) for each dataset and various regularization parameters. (BT did not satisfy (3.11) in the inex and usps experiments for $\lambda = 10^{-6}$ after 6000 iterations.)

3.4.3 Differentiable and Non-differentiable Risks

To examine the performance of BMRM on both differentiable and non-differentiable empirical risks, we performed binary classification experiments using hinge loss (3.9) and logistic loss:

$$\min_{\mathbf{w}} J(\mathbf{w}) := \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \log(1 + \exp(-y_i \langle \mathbf{w}, x_i \rangle)), \tag{3.12}$$

on the binary classification datasets mentioned in Section 3.4.1 with split similar to that in Section 3.4.1. Since there were other methods in the comparison which use different termination criteria, we compared their CPU time used in reducing the relative difference between the current smallest objective function value and the optimum:

$$\frac{\min_{i \in [t]} J(\mathbf{w}_i) - J(\mathbf{w}_*)}{J(\mathbf{w}_*)},$$

where $\mathbf{w}_i$ is the weight vector at time/iteration i, and $\mathbf{w}_*$ is the minimizer obtained by running BMRM until the approximation gap ϵ_t is less than or equal to 10^{-4}. The best $\lambda \in \{2^{-20}, \ldots, 2^0\}$ for each of the datasets was determined by evaluating the performance on the corresponding validation set.[13]

In the case of linear SVMs, we compared BMRM to three publicly available state-of-the-art batch solvers:

1. OCAS [Franc and Sonnenburg, 2008]. Since this method is equivalent to LSBMRM with binary hinge loss, we refer to this software by LSBMRM for naming consistency.
2. LIBLINEAR [Fan et al., 2008] version 1.33 with option "-s 3".
3. SVM$^{\text{perf}}$ [Joachims, 2006] version 2.5 with option "-w 3" and with double precision floating point numbers.

LIBLINEAR solves the dual problem of linear SVM using a coordinate descent method [Hsieh et al., 2008]. SVM$^{\text{perf}}$ was chosen for comparison as it is algorithmically identical to BMRM in this case. Both LIBLINEAR and SVM$^{\text{perf}}$ provide a "shrinking" technique to speed up the algorithms by ignoring some data points which are not likely to affect the objective. Since BMRM does not provide such shrinking technique, we excluded this option in both LIBLINEAR and SVM$^{\text{perf}}$ for a fair comparison.

Figure 3.8 shows the relative difference in objective value as a function of training time (CPU seconds) for three methods on various datasets. BMRM was faster than SVM$^{\text{perf}}$

[13] The corresponding penalty parameter C for LIBLINEAR and OCAS is $1/(m\lambda)$, and for SVM$^{\text{perf}}$ is $1/(100\lambda)$.

on all datasets except news20-b. The performance difference observed here was largely due to the differences in the implementations (*e.g.*, feature vector representation, QP solver, etc.). Nevertheless, both BMRM and SVM^{perf} were significantly outperformed by LSBMRM and LIBLINEAR on all datasets, and LIBLINEAR was almost always faster than LSBMRM. It is clear from the figure that LSBMRM and LIBLINEAR enjoy progression with "strictly" decreasing objective values; whereas the progress of both BMRM and SVM^{perf} were hindered by the "stalling" steps (*i.e.*, the flat line segments in the plots). The fact that LSBMRM is different from BMRM and SVM^{perf} by one additional line search step implies that the "stalling" steps were the time that BMRM and SVM^{perf} used to improve the approximation at the regions which did not directly contribute to reducing the primal objective function value.

In the case of logistic regression, we compared BMRM to the state-of-the-art trust region Newton method for logistic regression [Lin et al., 2008] which is also available in the LIBLINEAR package (option "-s 0"). From Figure 3.9, we see that LIBLINEAR outperformed BMRM on all datasets and that BMRM suffered from the same "stalling" phenomenon as observed in the linear SVMs case.

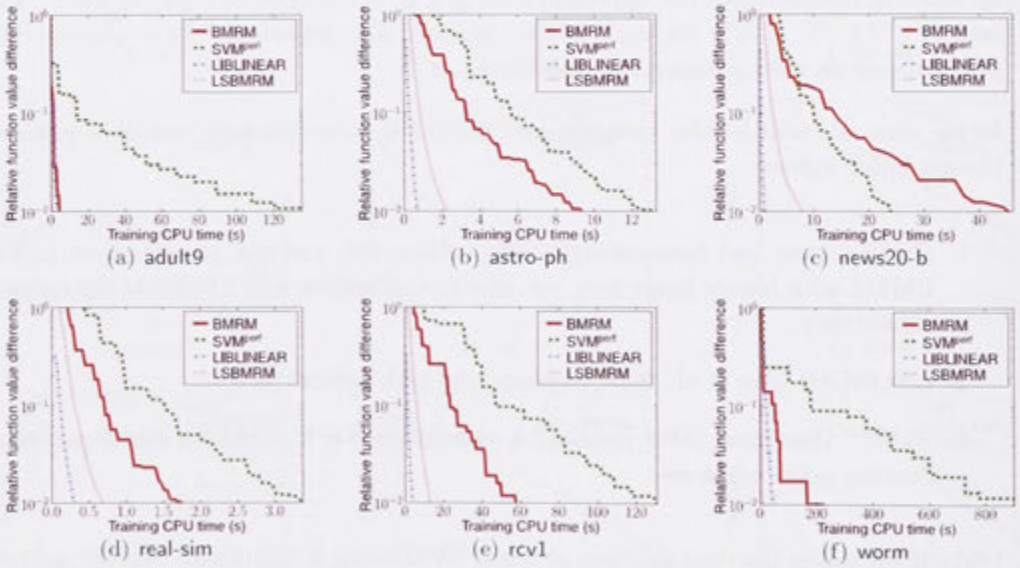


Figure 3.8: Linear SVMs. Relative primal objective value difference during training.

3.5 Related Work

Among the existing works on bundle methods for machine learning applications, the Structural SVMs (SVM^{struct}) [Tsochantaridis et al., 2005] and its variant SVM^{perf} [Joachims, 2006, Joachims et al., 2009] are the closest to ours. SVM^{struct} solves problems

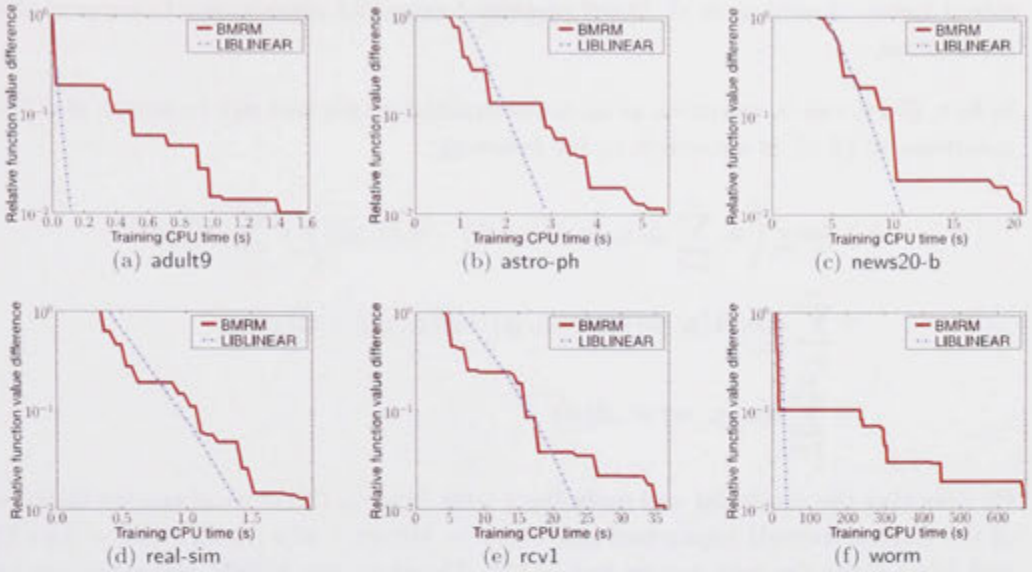


Figure 3.9: Logistic regression. Relative primal objective value difference during training.

with squared L_2 norm regularizer and max-margin empirical risk formulated primarily in a constrained optimization problem such as

$$(\mathbf{w}^m, \boldsymbol{\xi}^m) = \underset{\mathbf{w},\boldsymbol{\xi}}{\operatorname{argmin}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i \tag{3.13}$$

$$\text{s.t.} \quad \forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle \mathbf{w}, \phi(x_i, y_i) - \phi(x_i, y) \rangle \geq 1 - \frac{\xi_i}{\Delta(y_i, y)}, \ \xi_i \geq 0.$$

For the case where $\phi(x, y) \in \mathbb{R}^d$, instead of using m-slack as in SVM$^{\text{struct}}$, Joachims [2006] introduced the 1-slack formulation (SVM$^{\text{perf}}$) which aggregates constraints subject to all $i \in [m]$ and all $y \in \mathcal{Y}$. The resulting formulation reads

$$(\mathbf{w}^1, \xi^1) = \underset{\mathbf{w},\xi \geq 0}{\operatorname{argmin}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \xi \tag{3.14}$$

$$\text{s.t.} \quad \forall (\bar{y}_1, \ldots, \bar{y}_m) \in \mathcal{Y}^m :$$

$$\left\langle \mathbf{w}, \sum_{i=1}^{m} \Delta(y_i, \bar{y}_i)\left[\phi(x_i, y_i) - \phi(x_i, \bar{y}_i)\right] \right\rangle \geq \sum_{i=1}^{m} \Delta(y_i, \bar{y}_i) - \xi.$$

It has been shown [Joachims, 2006, Joachims et al., 2009] that solving both (3.13) and (3.14) arrive at the same solution *i.e.*, $\mathbf{w}^m = \mathbf{w}^1$ with $m^{-1}\|\boldsymbol{\xi}^m\|_1 = \xi^1$. By the cutting-plane method, (3.13) generates m constraints per iteration while (3.14) generates only one. Although the computational cost per iteration is the same in both methods, the resulting intermediate problem of (3.14) has m times less constraints hence it can be

solved faster. Joachims et al. [2009] presented empirical experiments to support this observation.

In fact, (3.14) can be rewritten as an unconstrained regularized risk by noting that the constraint of (3.14) is equivalent to the following:

$$\begin{aligned}
\xi &\geq \max_{\bar{\mathbf{y}} \in \mathcal{Y}^m} \left\langle \mathbf{w}, \sum_{i=1}^{m} \Delta(y_i, \bar{y}_i) \left[\phi(x_i, \bar{y}_i) - \phi(x_i, y_i)\right] \right\rangle + \sum_{i=1}^{m} \Delta(y_i, \bar{y}_i) \\
&= \sum_{i=1}^{m} \max_{\bar{y}_i \in \mathcal{Y}} \Delta(y_i, \bar{y}_i) \left\langle \mathbf{w}, \phi(x_i, \bar{y}_i) - \phi(x_i, y_i) \right\rangle + \Delta(y_i, \bar{y}_i) \\
&=: \sum_{i=1}^{m} l(x_i, y_i, \mathbf{w}) =: R(\mathbf{w}).
\end{aligned}$$

By removing the constraint and replacing ξ with $R(\mathbf{w})$ in (3.14) we obtain an instance of the (unconstrained) regularized risk $J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w})$ with $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ and $R(\mathbf{w})$ being the max-margin loss (2.14). Therefore, the BMRM algorithm can be seen as a generalization of SVM$^{\text{perf}}$.

3.6 Summary and Discussion

We have introduced a bundle method specialized to the regularized risk minimization (BMRM) and provided convergence analysis to show that BMRM has better theoretical convergence rate than the standard bundle method in this setting. We have also shown that BMRM inherits many properties of the standard bundle methods such as generality, modularity, amenability to parallel and distributed computing, and memory efficiency.

The algorithms and their convergence behavior we have presented in this chapter were restricted to the setting where the regularization parameter λ and empirical risk are fixed. Typically, a machine learning process involves multiple tries of regularization parameter in order to get the *best* solution for a fixed empirical risk and training dataset. We will show in the next chapter that BMRM is very efficient in this scenario, *i.e.*, repeatedly solving a regularized risk for different values of λ.

Chapter 4

Efficient Computation of Regularization Path

A typical machine learning process involves building many models and selecting the best for a problem at hand. In this chapter, we show that BMRM helps speed up the model selection problem significantly.

4.1 Introduction

In regularized risk minimization, model selection refers to the selection of a compatibility function space $\mathcal{F}$ from which the best compatibility function achieves the lowest generalization error (*i.e.*, risk). Since we are concerned with the case where $\mathcal{F} = \left\{\mathbf{w} \in \mathbb{R}^d \ : \ \|\mathbf{w}\| \leq \tau, \ \tau > 0\right\}$, the selection of $\mathcal{F}$ is essentially equivalent to the selection of τ (*cf.* Section 2.1.3). The selection of τ can, in turn, be reformulated as the selection of regularization parameter λ in the regularized risk minimization:

$$\min_{\mathbf{w}} J(\mathbf{w}; \lambda) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w}).$$

4.1.1 Regularization Path

A regularization path is a curve $r : (0, \infty) \rightarrow \mathbb{R}^d$ of weight vectors defined by

$$r(\lambda) = \operatorname*{argmin}_{\mathbf{w}} J(\mathbf{w}; \lambda).$$

Tracing $r(\lambda)$ over an interval $[\lambda_{\min}, \lambda_{\max}] \subset (0, \infty)$ means solving $J(\mathbf{w}; \lambda)$ for all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. The main purpose of tracing $r(\lambda)$ is to select the best λ such that the weight vector obtained after minimizing the corresponding $J(\mathbf{w}; \lambda)$ leads to the best performance on the hold-out validation set; this process is known as cross-validation [Duda et al., 2001]. Apart from model selection, the traced regularization path $r(\lambda)$ also

allows one to study the evolution of the feature weights (*i.e.*, the weight vector $\mathbf{w}$) as a function of the regularization parameter. Figure 4.1 illustrates the abovementioned two uses of regularization path for the case of training binary classifier using linear SVMs on the diabetes_scale[1] dataset.

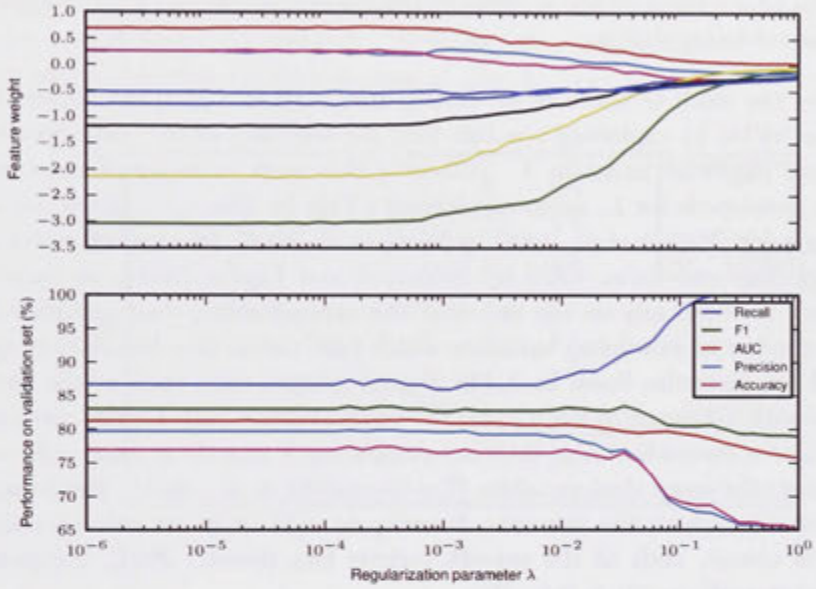


Figure 4.1: Tracing the regularization path $r(\lambda)$ of linear SVMs on the diabetes_scale dataset. **Top:** The evolution of 8 feature weights (*i.e.*, weight vector $\mathbf{w} \in \mathbb{R}^8$) trained on 70% of diabetes_scale with different values of λ. **Bottom:** Performance on validation set (Recall, F1, AUC, Precision, and Accuracy) of classifiers with feature weights trained with different values of λ, on the remaining 30% of diabetes_scale.

4.1.2 Existing Regularization Path Tracing Algorithms

Generally, the methods for tracing $r(\lambda)$ can be categorized as either exact or approximate. By exploiting the fact that $r(\lambda)$ is piecewise linear for regularized risks such as SVMs and its variants, exact methods manage to compute the curve $r(\lambda)$ exactly for all λ in a given interval in time comparable to that of minimizing $J(\mathbf{w}; \lambda)$ for just a single λ [Hastie et al., 2004]. Efron et al. [2004] pioneered the field of exact regularization path tracing by presenting one such method for solving the L_1 norm regularized least

[1] diabetes_scale consists of 768 data points with 8 features and is available for download at LIBSVM tools website http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

squares regression:

$$\min_{\mathbf{w}} \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^{m} \left\| y_i - \mathbf{w}^\top x_i \right\|^2, \tag{4.1}$$

where $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, for all $\lambda \geq 0$. Briefly, the efficient algorithm developed by Efron et al. [2004] finds all the d joints of the curve $r(\lambda)$ for (4.1) and completes the curve by linear interpolation.

Inspired by the work of Efron et al. [2004], Hastie et al. [2004] developed an exact method for SVMs by exploiting the fact that the variables of the corresponding dual problem are piecewise linear in λ. Following this work, a series of exact methods have been developed: for L_1 norm regularized SVMs by Zhu et al. [2004]; for support vector regression [Vapnik et al., 1997] by Wang et al. [2006]; for support vector domain description [Tax and Duin, 1999] by Sjöstrand and Larsen [2006], to name a few. These exact methods rely on the fact that the corresponding dual problems have m (*i.e.*, the number of examples) variables which take values in a bounded range (*e.g.*, $[0, 1]$) and are piecewise linear in λ. In a more general case, such as the structured prediction with max-margin loss (2.14), the regularization path for the corresponding dual problem is piecewise linear in λ but computing it exactly is intractable as there are exponentially many dual variables [Tsochantaridis et al., 2005]. Furthermore, for regularized risks where the piecewise linearity in $r(\lambda)$ or in its corresponding dual variables is absent, such as the smooth logistic loss [Rosset, 2004], computing the regularization path exactly is infeasible.

One alternative to consider is an approximation $\hat{r}(\lambda)$ of $r(\lambda)$ which can be generated by solving $J(\mathbf{w}; \lambda)$ exactly for a finite set of λ and then completing the curve by linear interpolation. Rosset [2004] presented an incremental Newton-Raphson method which minimizes a smooth $J(\mathbf{w}; \lambda)$ starting with a small $\lambda_{\min} > 0$ then increases λ by a small value $\delta > 0$ at each iteration until the larger $\lambda_{\max}$ is reached. Under some regularity conditions, the difference between the resulting approximate path $\hat{r}(\lambda)$ and the true path $r(\lambda)$ is guaranteed [Rosset, 2004] to be in the order of $O(\delta^2)$, *i.e.*,

$$\forall c \in [\lambda_{\min}, \lambda_{\max}] : \ \|\hat{r}(c) - r(c)\| = O(\delta^2).$$

In general, computing the approximate regularization path involves solving $J(\mathbf{w}; \lambda)$ for a sequence of regularization parameters $\lambda_1 > \cdots > \lambda_k$. By continuity, a small change in the value of λ will only result in slight displacement of the minimizer of $J(\mathbf{w}; \lambda)$. To better see this observation, we illustrate in Figure 4.2 a 1-dimensional convex quadratic objective:

$$J(w; \lambda) = \lambda \underbrace{(w^2/2)}_{\Omega(w)} + \underbrace{(w^2 - 8w + 20)}_{R(w)},$$

for $\lambda \in \{1, 0.5, 0.1, 0.01\}$. From the figure, we see that the minimizers of $J(w; \lambda)$, $\lambda = 1, 0.5, 0.1, 0.01$ (indicated by vertical dashed lines) tend to the minimizer of R as λ is decreased. Furthermore, the minimizers of $J(\mathbf{w}; \lambda)$ for different λ are located adjacent to those with adjacent values of λ. This observation is in line with the commonly seen "warm"-start strategy that initializes an optimization with a point *believed* to be close to the optimum. We see in the next section that BMRM allows "hot"-starting (*i.e.*, a better strategy than warm-starting) based on the fact that the approximation $\check{R}_t$ remains valid when λ is changed. Therefore, the approximation $\check{R}_t$ can be reused throughout the sequence of minimization of $J(\mathbf{w}; \lambda)$, $\lambda = \lambda_1, \ldots, \lambda_k$.

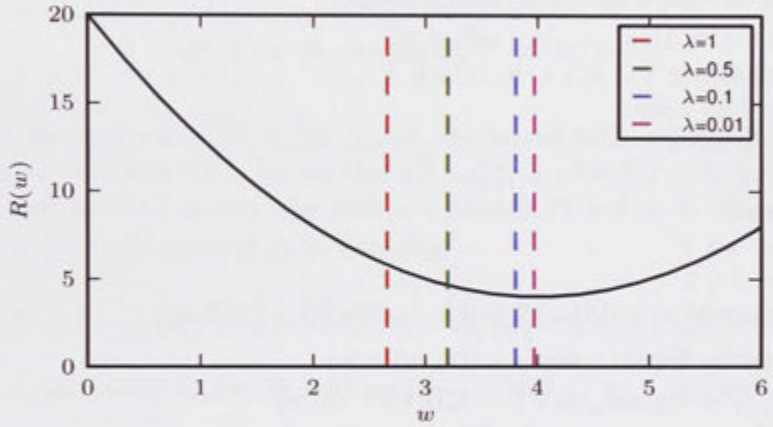


Figure 4.2: Minimizers (vertical dashed lines) of $J(w; \lambda)$, $\lambda = 1, 0.5, 0.1, 0.01$, where $w \in \mathbb{R}, \Omega(w) = w^2/2$, and $R(w) = w^2 - 8w + 20$.

4.2 BMRM for Regularization Path

We describe in this section how BMRM is effectively used for computing approximate regularization path. We also present iteration bounds for two common sequences of regularization parameters, namely, arithmetic and geometric.

4.2.1 Algorithm

In standard bundle methods, the approximation $\check{J}_t$ of the regularized risk $J(\mathbf{w}; \lambda) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w})$ is invalidated when the regularization parameter λ is changed. This is because $J(\mathbf{w}; \lambda_1) \neq J(\mathbf{w}; \lambda_2)$ for $\lambda_1 \neq \lambda_2$ and the linearizations of $J(\mathbf{w}; \lambda_1)$ does not, in general, hold for $J(\mathbf{w}; \lambda_2)$.

Unlike the standard bundle methods, BMRM approximates only R via

$$\check{R}_t(\mathbf{w}) := \max_{i \in [t]} \{\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i\}.$$

This approximation is independent of the regularization parameter λ. This means that the approximation $\check{R}_t^{\lambda_1}$ of R built for the minimization of $J(\mathbf{w}; \lambda_1)$ can be used to initialize the minimization of $J(\mathbf{w}; \lambda_2)$. The reuse of approximation is done sequentially until all minimizations of $J(\mathbf{w}; \lambda_i)$, $i \in [k]$ are done. Algorithm 6 lists the details of the procedure for minimizing a sequence of $J(\mathbf{w}; \lambda_i)$, $i \in [k]$.

Algorithm 6 BMRM for regularization path

input: $\epsilon > 0$, $\lambda_1 > \cdots > \lambda_k$, $\mathbf{w}_*^{\lambda_0} \in \mathbb{R}^d$
initialization: $s \leftarrow 0$, $t \leftarrow 0$, $W = \emptyset$
for $p = 1$ **to** k **do**
  $\mathbf{w}_1^{\lambda_p} \leftarrow \mathbf{w}_*^{\lambda_{p-1}}$
  $s \leftarrow 0$
  repeat
    $s \leftarrow s + 1$
    $t \leftarrow t + 1$
    Compute $\mathbf{a}_t \in \partial R(\mathbf{w}_s^{\lambda_p})$ and $b_t \leftarrow R(\mathbf{w}_s^{\lambda_p}) - \left\langle \mathbf{w}_s^{\lambda_p}, \mathbf{a}_t \right\rangle$
    Update $\check{R}_t(\mathbf{w}) := \max_{i \in [t]} \{\langle \mathbf{w}, \mathbf{a}_i \rangle + b_i\}$
    $\mathbf{w}_{s+1}^{\lambda_p} \leftarrow \operatorname{argmin}_{\mathbf{w}} J_t(\mathbf{w}) := \lambda_p \Omega(\mathbf{w}) + \check{R}_t(\mathbf{w})$
    $\mathbf{w}_*^{\lambda_p} := \operatorname{argmin}_{q \in [s+1]} J(\mathbf{w}_q^{\lambda_p})$
    $\epsilon_s^{\lambda_p} \leftarrow J(\mathbf{w}_*^{\lambda_p}) - J_t(\mathbf{w}_{s+1}^{\lambda_p})$
  until $\epsilon_s \leq \epsilon$
  $W \leftarrow W \cup \left\{ \mathbf{w}_*^{\lambda_p} \right\}$
end for
return W

Clearly, Algorithm 6 has Algorithm 2 as an inner loop. Therefore, the implementation of Algorithm 6 is easily obtained by minor modification to that of Algorithm 2. Furthermore, the properties of BMRM such as amenability to parallel/distributed computation and linearization selection/aggregation are retained in this modification.

4.2.2 Iteration Bounds

We give a straightforward theorem which bounds the number of iterations required to minimize $J(\mathbf{w}; c\lambda)$, $c \in (0, 1)$ when BMRM is initialized with the approximation $\check{R}_t$ and solution $\mathbf{w}_*^{\lambda}$ obtained after $J(\mathbf{w}; \lambda)$ is minimized. The assumptions on the regularized risk follow that in Chapter 3 *i.e.*, the regularizer Ω is σ-strongly convex and $\sup_i \|\mathbf{a}_i\| \leq G$ for all $\mathbf{a}_i \in \partial R(\mathbf{w}_i)$. For generality, we assume only that R is convex and locally Lipschitz continuous but not necessarily differentiable.

Theorem 4.2.1. *Given any $\lambda > 0$, $c \in (0, 1)$ and $\epsilon < 4G^2/(\lambda\sigma)$ Assume that* BMRM *minimized $J(\mathbf{w}; \lambda)$ to precision ϵ after T_λ iterations, i.e. $\epsilon^\lambda_{T_\lambda} \leq \epsilon$, and returned the final approximation $\check{R}_{T_\lambda}$ and solution $\mathbf{w}^\lambda_*$. With initial approximation $\check{R}_{T_\lambda}$ and initial point $\mathbf{w}^{c\lambda}_1 := \mathbf{w}^\lambda_*$, the number of iterations* BMRM *takes to minimize $J(\mathbf{w}; c\lambda)$ to the same precision ϵ is at most*

$$T_{c\lambda} \leq \begin{cases} \frac{8G^2}{c\lambda\sigma}\left(\frac{1}{\epsilon} - \frac{1}{\epsilon^{c\lambda}_1}\right) + 1 & \text{if } \epsilon^{c\lambda}_1 \leq 4G^2/(c\lambda\sigma) \\ \log_2\left(\frac{c\lambda\sigma\epsilon^{c\lambda}_1}{4G^2}\right) + \frac{8G^2}{c\lambda\sigma\epsilon} - 1 & \text{if } \epsilon^{c\lambda}_1 > 4G^2/(c\lambda\sigma) \end{cases}$$

where $\epsilon^{c\lambda}_1 := \min_{1\leq i\leq 2} J(\mathbf{w}^{c\lambda}_i; c\lambda) - J_{T_\lambda}(\mathbf{w}^{c\lambda}_2; c\lambda)$. Furthermore,

$$\epsilon^{c\lambda}_1 \leq \epsilon + (\lambda - c\lambda)[\Omega(\mathbf{w}^{c\lambda}_2) - \Omega(\mathbf{w}^{c\lambda}_1)]. \tag{4.2}$$

Proof. For the case where $\epsilon^{c\lambda}_1 > 4G^2/(c\lambda\sigma)$, the bound follows from Theorem 3.2.2. Otherwise, by Theorem 3.2.1, we see that $\epsilon^{c\lambda}_t - \epsilon^{c\lambda}_{t+1} \geq c\lambda\sigma(\epsilon^{c\lambda}_t)^2/(8G^2)$ for all $t > 0$. Applying Lemma B.1.1 in this case with $z = c\lambda\sigma/(8G^2)$ and $p_1 = \epsilon^{c\lambda}_1$ we get $T_{c\lambda} \leq \frac{8G^2}{c\lambda\sigma}\left(\frac{1}{\epsilon} - \frac{1}{\epsilon^{c\lambda}_1}\right) + 1$. To prove (4.2) we note that

$$\epsilon^{c\lambda}_1 \leq J(\mathbf{w}^{c\lambda}_1; c\lambda) - J_{T_\lambda}(\mathbf{w}^{c\lambda}_2; c\lambda) = J(\mathbf{w}^{c\lambda}_1; \lambda) - J_{T_\lambda}(\mathbf{w}^{c\lambda}_2; \lambda) + (\lambda - c\lambda)[\Omega(\mathbf{w}^{c\lambda}_2) - \Omega(\mathbf{w}^{c\lambda}_1)].$$

Since $\mathbf{w}^\lambda_{T_\lambda+1}$ minimizes $J_{T_\lambda}(\mathbf{w}; \lambda)$, we have that $J_{T_\lambda}(\mathbf{w}^\lambda_{T_\lambda+1}; \lambda) \leq J_{T_\lambda}(\mathbf{w}^{c\lambda}_2; \lambda)$. Hence,

$$J(\mathbf{w}^{c\lambda}_1; \lambda) - J_{T_\lambda}(\mathbf{w}^{c\lambda}_2; \lambda) \leq J(\mathbf{w}^{c\lambda}_1; \lambda) - J_{T_\lambda}(\mathbf{w}^\lambda_{T_\lambda+1}; \lambda) = \epsilon^\lambda_{T_\lambda} \leq \epsilon$$

and we have completed the proof. □

Theorem 4.2.1 implies that hot-starting the minimization of $J(\mathbf{w}; c\lambda)$ with the approximation $\check{R}_{T_\lambda}(\mathbf{w})$ and the solution $\mathbf{w}^\lambda_*$ obtained in the minimization of $J(\mathbf{w}; \lambda)$ would incur only a small number of iterations when c is sufficiently close to 1. This observation is reasonable as $J_t(\mathbf{w}; \lambda)$ is continuous in λ therefore slight changes in λ will not affect the accuracy of the approximation much.

By summing the results of Theorem 4.2.1 over a sequence of k regularization parameters: $\lambda_1 > \lambda_2 > \ldots > \lambda_k$, we obtain the best case T_{best} and worst case T_{worst} upper bounds of the total number of iterations required in minimizing the sequences of problems $J(\mathbf{w}; \lambda_i)$ for all $i \in [k]$. The best case refers to the scenario where the regularization parameters λ_i are close to each other (and similarly the approximations $\check{R}^{\lambda_i}_t$), such that $\epsilon^{\lambda_i}_1$ are less than $4G^2/(\lambda_1\sigma)$ for all $i \in [k]$. On the contrary, the worst case refers to the scenario where $\epsilon^{\lambda_i}_1 > 4G^2/(\lambda_1\sigma)$ for all $i \in [k]$; this bound can be simply stated as k times the upper bound for the smallest regularization parameter λ_k. We state the bounds for the two cases in the following corollary.

Corollary 4.2.2. *Let $\lambda_1 > \lambda_2 > \ldots > \lambda_k$ be a sequence of positive numbers. The best case T_{best} and worst case T_{worst} upper bounds of total number of iterations required by*

Algorithm 6 to minimize the sequence of problems: $J(\mathbf{w};\lambda_i)$ for all $i \in [k]$, to precision $\epsilon < 4G^2/(\lambda_1\sigma)$ in sequential order are as follow:

$$T_{\text{best}} = T_{\lambda_1} + 8G^2\sigma^{-1}\sum_{i=2}^{k}\frac{1}{\lambda_i}\left(\frac{1}{\epsilon} - \frac{1}{\epsilon_1^{\lambda_i}}\right) + k - 1 \tag{4.3}$$

$$\leq \log_2\left(\frac{\lambda_1\sigma\epsilon_1^{\lambda_1}}{4G^2}\right) + \frac{8G^2}{\sigma\epsilon}\sum_{i=1}^{k}\frac{1}{\lambda_i} - k \quad [\textit{by replacing } \epsilon_1^{\lambda_i} \textit{ with } 4G^2/(\lambda_i\sigma)] \tag{4.4}$$

$$T_{\text{worst}} = k\log_2\left(\frac{\lambda_k\sigma\epsilon_1^{\lambda_k}}{4G^2}\right) + \frac{8G^2k}{\lambda_k\sigma\epsilon} - k$$

More concretely, we specialize the best case bound in Corollary 4.2.2 for two common types of sequence of regularization parameters: arithmetic and geometric sequences. The following theorems state the specialized results which highlight the influence of the choice of k, λ_1, and λ_k to the bounds.

Theorem 4.2.3. *For a given arithmetic sequence $\lambda_i = \lambda_1 - (i-1)c > 0$, for all $i \in [k]$ and for some $c > 0$. The best case bound T_{best} in Corollary 4.2.2 can be restated as follows:*

$$T_{\text{best}} \leq \log_2\left(\frac{\lambda_1\sigma\epsilon_1^{\lambda_1}}{4G^2}\right) + \frac{8G^2}{\sigma\epsilon}\left(\frac{1}{\lambda_k} + \frac{k-1}{\lambda_1 - \lambda_k}\log\left(\frac{\lambda_1}{\lambda_k}\right)\right) - k$$

Proof. To prove the upper bounds of T_{best} we just need to simplify and upper-bound the term $\sum_{i=1}^{k}\lambda_i^{-1}$. Note that $\sum_{i=1}^{k}\lambda_i = k(\lambda_1+\lambda_k)/2$ and $c(k-1) = \lambda_1 - \lambda_k$. So, we have that,

$$\begin{aligned}\sum_{i=1}^{k}\frac{1}{\lambda_i} &= \sum_{i=1}^{k}\frac{1}{\lambda_k + (i-1)c} = \frac{1}{c}\sum_{i=1}^{k}\frac{1}{\lambda_k/c + i - 1} = \frac{1}{c}\sum_{i=\frac{\lambda_k}{c}}^{k-1+\frac{\lambda_k}{c}}\frac{1}{i} \\ &\leq \frac{1}{c}\left(\frac{c}{\lambda_k} + \int_{\frac{\lambda_k}{c}}^{k-1+\frac{\lambda_k}{c}}\frac{1}{i}di\right) = \frac{1}{\lambda_k} + \frac{1}{c}\log\left(\frac{k-1+\lambda_k/c}{\lambda_k/c}\right) \\ &= \frac{1}{\lambda_k} + \frac{\lambda_1 - \lambda_k}{k-1}\log\left(\frac{\lambda_1}{\lambda_k}\right).\end{aligned}$$

Substituting the result into (4.4) we proved the claim. □

We now show the specialized best case bound for geometric sequence of regularization parameters.

Theorem 4.2.4. *For a given geometric sequence $\lambda_i = \lambda_1 c^{i-1} > 0$, for all $i \in [k]$ and for some $c \in (0,1)$. The best case bound T_{best} in Corollary 4.2.2 can be restated as the*

following:

$$T_{\text{best}} \leq \log_2 \left(\frac{\lambda_1 \sigma \epsilon_1^{\lambda_1}}{4G^2} \right) + \frac{8G^2}{\sigma\epsilon} \left(\frac{\lambda_k^{-1} - \lambda_1^{-1}}{1 - \sqrt[k-1]{\lambda_k/\lambda_1}} \right) - k$$

Proof. Observe that the sum of a geometric sequence is

$$\sum_{i=1}^{k} az^{i-1} = \frac{a(z^k - z)}{z - 1} = \frac{az^{k-1} - a}{1 - 1/z}$$

for $z \neq 1$. Therefore, we have

$$\sum_{i=1}^{k} \frac{1}{\lambda_i} = \sum_{i=1}^{k} \lambda_1^{-1} (c^{-1})^{i-1} = \frac{\lambda_k^{-1} - \lambda_1^{-1}}{1 - c} = \frac{\lambda_k^{-1} - \lambda_1^{-1}}{1 - \sqrt[k-1]{\lambda_k/\lambda_1}}$$

by substituting $c = \sqrt[k-1]{\lambda_k/\lambda_1}$ which is derived from $\lambda_k = \lambda_1 c^{k-1}$. Substituting the results into (4.4) we proved the claim. □

A few observations are in order: Firstly, we see that both examples of regularization parameter sequences have inverse dependence on the smallest regularization parameter λ_k when λ_1 and k are fixed. This is not surprising as Algorithm 6 is just a series of executions of Algorithm 2. Secondly, the best case bound for arithmetic sequence of λ_i has linear dependence on k compared to that of the geometric sequence which has exponential dependence on k. Nevertheless, these best case bounds are rather loose as the difference between adjacent regularization parameters *i.e.*, $\lambda_i - \lambda_{i+1}$ which influences the first approximation gaps $\epsilon_1^{\lambda_i}$ (and hence the terms $(\epsilon^{-1} - (\epsilon_1^{\lambda_i})^{-1})$ in (4.3)) are not considered in these bounds; we show the effects of the difference $\lambda_i - \lambda_{i+1}$ in the numerical experiments in next section.

4.3 Experiments

4.3.1 Experimental Setup

In this section, we study empirically the efficacy of hot-start strategy, *i.e.*, Algorithm 6, in computing the approximate regularization path for training binary classifiers with

the following empirical risks:

$$\text{hinge:} \quad R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i \langle \mathbf{w}, x_i \rangle)$$

$$\text{squared hinge:} \quad R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i \langle \mathbf{w}, x_i \rangle)^2$$

$$\text{logistic:} \quad R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} \log(1 + \exp(-y_i \langle \mathbf{w}, x_i \rangle))$$

in combination with the squared L_2 norm regularizer $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. The datasets we used in the experiments are adult9, colon-cancer, duke breast-cancer, leukemia, mushrooms, rcv1, real-sim, splice, sonar, and web8. These datasets are available on the LIBSVM tools website[2]. Table 4.1 summarizes the properties of these datasets.

Dataset	#examples m	dimension d	density (%)
adult9	48,842	123	11.27
colon	62	2,000	100.00
duke	38	7,129	100.00
leukemia	38	7,129	100.00
mushrooms	8,124	112	18.75
rcv1	20,242	47236	0.15
real-sim	72,201	20,958	0.25
sonar	208	60	100.00
splice	1,000	60	100.00
web8	45,546	300	12.73

Table 4.1: Properties of the binary classification datasets used in the experiments.

We set $\lambda_1 = 10^{-2}$ and $\lambda_k = 10^{-6}$ as the best λ for most of the datasets fall in this range. On each of the datasets and with each of the empirical risks, we solved the sequence of problems $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$, $i \in [k]$ for $k \in \{10, 20, 50, 100\}$ and for both arithmetic and geometric sequences of λ_i. For each of the experiments, we terminated the algorithms when the termination criterion is satisfied:

$$J(\mathbf{w}_*^{\lambda_i}; \lambda_i) - J_t(\mathbf{w}_{t+1}^{\lambda_i}; \lambda_i) \leq \epsilon J(\mathbf{w}_*^{\lambda_i}; \lambda_i),$$

where $\epsilon = 10^{-2}$.

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

4.3.2 Experimental Results

We compared the hot-start strategy with the warm-start and cold-start strategies. Warm-start strategy initializes the problem $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$ with only the solution $\mathbf{w}_*^{\lambda_{i-1}}$ obtained after solving $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_{i-1})$. Cold-start strategy solves the problems $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$, $i \in [k]$ independent of each other and use neither previous solution nor previous approximation for initialization.

Table 4.2 shows the speedup of the hot-start strategy over the warm-start strategy in computing the regularization path for the three abovementioned regularized risks with 10, 20, 50, and 100 regularization parameters of arithmetic and geometric sequences in the range $[10^{-6}, 10^{-2}]$. Since warm-start strategy outperformed the cold-start strategy in most of cases, we do not include the results of cold-start strategy here. We see significant speedup in using the hot-start strategy for all three regularized risks as k is increased. In particular, the speedup in the case of geometric sequence of λ_i is more apparent. This is because most of the λ_i are distributed in a small region close to the end parameter $\lambda_k = 10^{-6}$ because of which the warm-start and cold-start strategies to take longer number of iterations to solve $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$ for each λ_i. Furthermore, the difference between adjacent regularization parameter (*i.e.*, $\lambda_i - \lambda_{i+1}$) decreases exponentially fast towards λ_k, hence, by Theorem 4.2.1, we know that hot-start strategy will require only small number of additional iterations to solve the problems $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$, $i \in [k]$ in sequential order.

To illustrate the influence of the difference $\delta_i := \lambda_i - \lambda_{i+1}$ to the efficacy of hot-start strategy more closely, we plot the number of iterations required by the hot-start strategy to solve each of the problems $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$ as a function of λ_i. Figure 4.3 and 4.4 show the numbers of iterations used in solving linear SVMs (*i.e.*, squared L_2 norm regularizer and hinge empirical risk) on all datasets, with $k = 10$ and $k = 100$, for the cases of arithmetic and geometric sequences of λ_i, respectively.

From Figure 4.3 we see that the advantage of smaller difference δ_i, *i.e.*, the case of $k = 100$, is not significant compared to the case of larger δ_i, *i.e.*, $k = 10$, when λ_i are larger than 10^{-3}. The difference in the number of iterations per λ is tiny – one explanation for this observation is that these λ_i are relatively large and hence the inverse scaling of λ does not dominate the convergence of BMRM (yet). For λ_i smaller than 10^{-3}, the plots show a sharp increase in the number of iterations, that is, the inverse scaling of λ started to take effect, so, the advantage of smaller δ_i is more apparent.

From Figure 4.4 we see that the difference in the number of iterations per each λ between the cases of $k = 10$ and $k = 100$ is significant over the whole sequence λ_i, regardless the exponential scaling of k stated in Theorem 4.2.4. The explanation for this empirical observation is that the difference δ_i is sufficiently small for most of the λ_i (due to the nature of geometric sequence), so, the minimizers of $J(\mathbf{w}; \lambda_i)$ and $J(\mathbf{w}; \lambda_{i+1})$ are *close* to each other. This implies that the approximation of R for $J(\mathbf{w}; \lambda_i)$ is

also *accurate* for $J(\mathbf{w}; \lambda_{i+1})$. We note that the efficacy of arithmetic and geometric sequences of λ_i cannot be compared directly in general as they refer to different values of λ_i (except λ_1 and λ_k).

4.4 Conclusion and Discussion

In this chapter, we have shown that BMRM is efficient in solving a series of regularized risk minimizations with slight change in the regularization parameters, which is a very common scenario in typical machine learning process. Also, this benefit of BMRM is unique as it is not (directly) observed in the standard bundle methods where the linearizations of the objective function are dependent on the regularization parameter.[3]

This chapter concludes the development of an efficient bundle methods for regularized risk minimization. In the next chapter, we develop a general framework of empirical risks which allows the incorporation of prior knowledge and local invariances of training examples to improve the outcome of learning.

[3]However, by storing the (sub)gradients of regularizer and empirical risk separately, the standard bundle methods can reuse the "modified" past linearizations for hot-starting, albeit using twice the memory space.

Dataset	Arithmetic				Geometric			
	$k = 10$	$k = 20$	$k = 50$	$k = 100$	$k = 10$	$k = 20$	$k = 50$	$k = 100$
adult9	1.31	1.68	3.20	4.87	8.78	15.37	34.57	57.76
	2.06	2.03	2.81	4.40	14.39	28.46	51.80	80.62
	1.17	1.62	2.49	3.78	8.82	14.13	32.54	53.91
colon	5.67	9.30	15.07	18.41	3.63	9.95	13.82	13.23
	2.49	4.02	8.92	16.60	4.66	7.83	21.53	36.51
	1.68	2.43	4.36	7.63	1.88	2.71	5.54	11.61
duke	6.00	9.04	12.39	14.55	3.21	11.69	14.78	15.60
	2.97	5.39	13.06	24.19	5.48	10.22	23.81	47.16
	1.80	2.77	4.75	7.15	2.17	2.81	6.12	10.53
leukemia	5.35	8.74	12.55	14.04	2.82	9.91	12.53	13.55
	2.93	5.77	12.78	26.91	5.51	10.38	24.89	44.57
	1.75	2.58	4.59	6.31	2.08	2.69	5.63	9.61
mushrooms	4.29	7.82	14.76	19.07	4.95	9.02	19.62	29.18
	1.48	2.14	3.82	5.39	2.79	5.25	10.80	17.74
	1.22	1.37	1.47	1.67	1.04	1.27	1.59	1.79
rcv1	0.97	0.97	1.07	1.08	1.11	1.51	2.45	3.54
	1.06	1.08	1.27	1.32	1.15	1.50	2.66	5.12
	1.02	1.09	1.08	1.12	1.13	1.59	2.74	3.89
real-sim	1.01	1.10	1.19	1.20	1.22	1.48	2.80	4.48
	1.02	1.05	1.07	1.27	1.24	1.50	2.98	4.98
	1.02	1.03	1.07	1.05	1.26	1.51	2.34	3.47
sonar	1.65	2.38	4.74	7.08	1.95	3.49	7.80	13.61
	1.24	1.59	2.81	4.54	2.91	6.85	15.22	23.84
	1.22	1.71	2.27	3.53	2.03	4.24	9.83	15.16
splice	5.00	8.13	16.20	22.97	37.15	63.77	107.57	186.92
	16.30	23.47	34.92	40.37	79.84	158.03	309.55	424.43
	17.04	19.35	19.11	18.64	96.75	139.53	231.77	272.75
web8	1.04	0.98	1.41	1.69	2.15	4.01	8.63	14.16
	0.93	1.17	1.14	1.59	4.35	9.01	17.73	30.69
	0.90	1.04	1.23	1.34	1.77	3.97	8.37	12.91

Table 4.2: Speedup of hot-start strategy in solving the problems $\min_{\mathbf{w}} J(\mathbf{w}; \lambda_i)$, $i \in \{\lambda_1 = 10^{-2}, \ldots, \lambda_k = 10^{-6}\}$ with $k \in \{10, 20, 50, 100\}$. Columns 2 through 5 and 6 through 9 indicate the ratios of the total number of iterations of warm-start strategy to that of hot-start strategy for arithmetic and geometric sequences of λ_i, respectively. The first, second and third rows corresponding to each dataset indicate the ratios obtained with hinge, squared hinge, and logistic empirical risks, respectively.

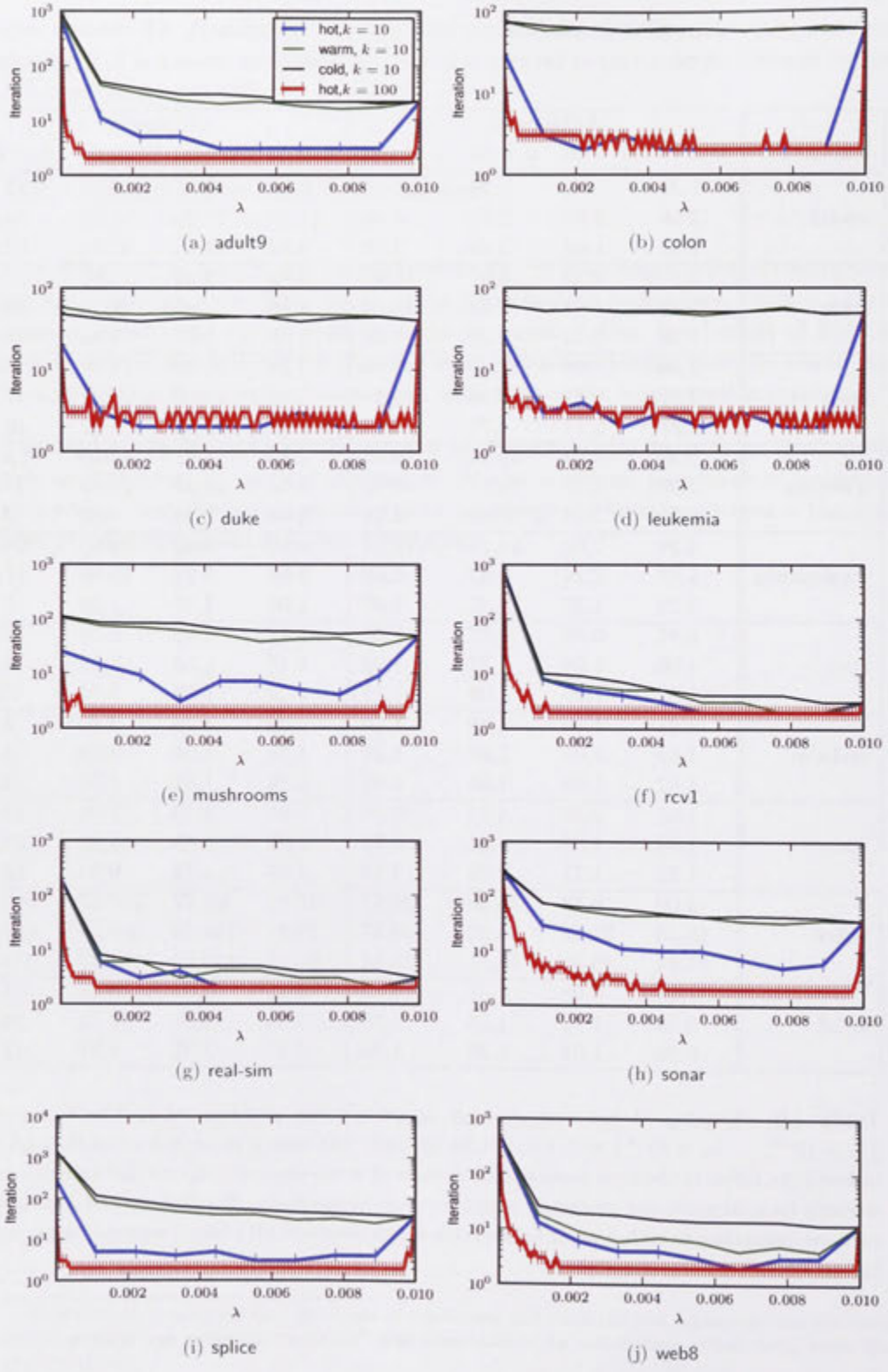
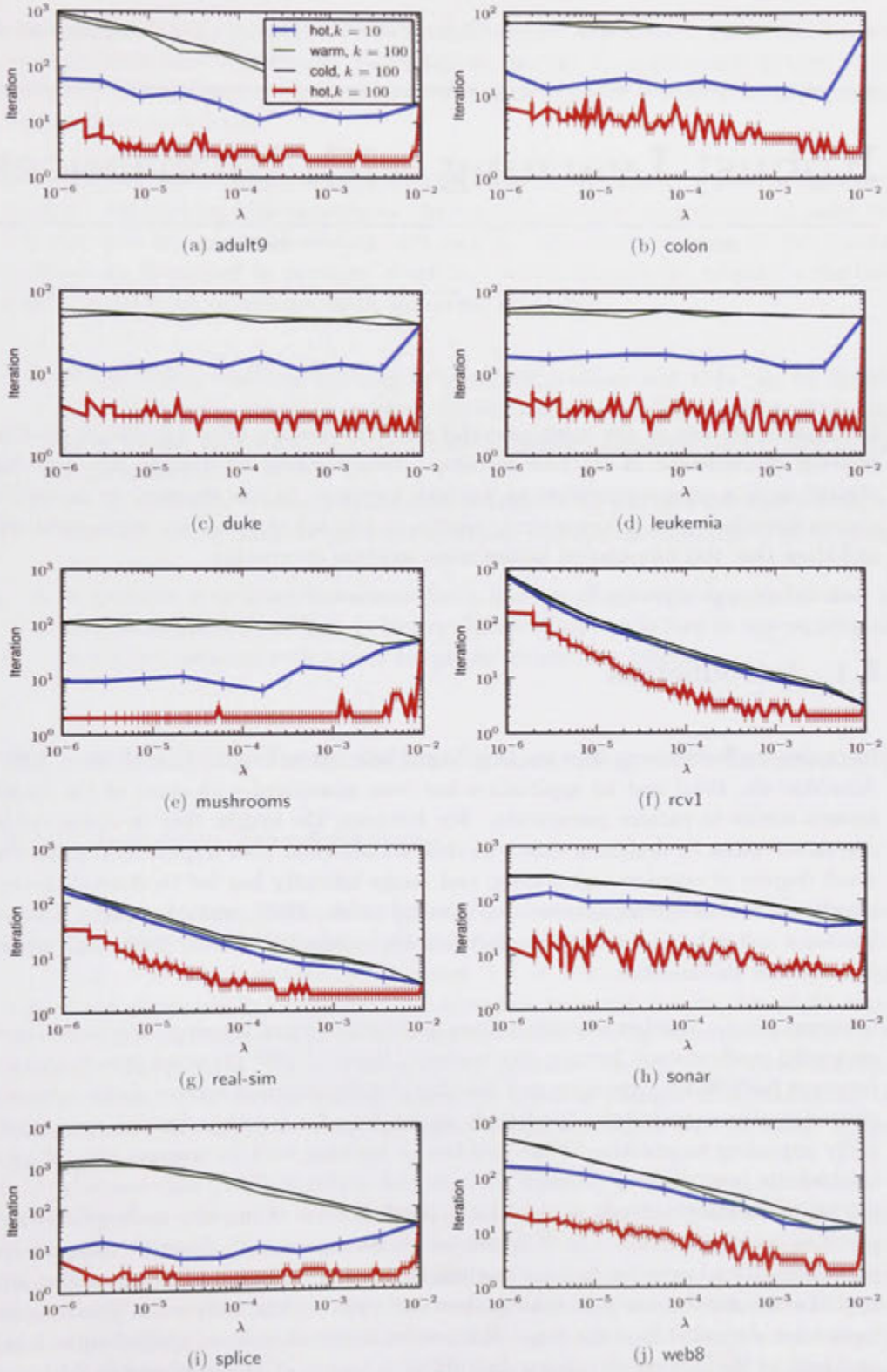


Figure 4.3: The number of iterations hot-start strategy used to compute the approximate regularization path for linear SVMs on various datasets with arithmetic sequence of λ_i and $k \in \{10, 100\}$. The baselines are the warm-start and cold-start strategies with $k = 10$.



(a) adult9 (b) colon

(c) duke (d) leukemia

(e) mushrooms (f) rcv1

(g) real-sim (h) sonar

(i) splice (j) web8

Figure 4.4: The number of iterations hot-start strategy used to compute the approximate regularization path for linear SVMs on various datasets with geometric sequence of λ_i and $k \in \{10, 100\}$. The baselines are the warm-start and cold-start strategies with $k = 10$.

Chapter 5

Robust Learning with Invariances

Invariances are one of the most powerful forms of domain prior knowledge used to improve the outcome of machine learning. Incorporating invariances into learning algorithms is a common problem in machine learning. In this chapter, we provide a convex formulation for incorporating invariances into arbitrary convex regularized risk and show that this formulation unifies many existing approaches.

5.1 Introduction

Incorporating invariances into learning algorithms has a long history [Hinton, 1987, Abu-Mostafa, 1992] and its application has been associated with some of the major success stories in pattern recognition. For instance, the insight that in vision tasks, one should often be designing detectors that are invariant with respect to translation, small degrees of rotation and scaling, and image intensity has led to state-of-the-art algorithms including tangent-distance [Simard et al., 1993], virtual support vectors [DeCoste and Schölkopf, 2002], group theoretic approach [Kondor, 2008], and others [Ferraro and Caelli, 1994].

In recent years a number of authors have attempted to put learning with invariances on a solid mathematical footing. For instance, Burges [1999] discusses how to extract invariant features for estimation and learning *globally* invariant estimators for a known class of invariance transforms (preferably arising from Lie groups). Another mathematically appealing formulation of the problem of learning with invariances casts it as a semidefinite programming problem [Graepel and Herbrich, 2004]; unfortunately this is neither particularly efficient to implement (having worse than cubic scaling behavior) nor does it cover a wide range of invariances in an automatic fashion. A different approach has been to pursue "robust" estimation methods which, roughly speaking, aim to find estimators whose performance does not suffer significantly when the observed inputs are degraded in some way. Robust estimation has been applied to learning problems in the context of missing data [Bhattacharyya et al., 2005] and to deal with

specific type of data corruption at test time [Globerson and Roweis, 2006]. The former approach leads to a second order cone program, limiting its applicability to very small datasets; the latter is also computationally demanding and is limited to only specific types of data corruption.

The goal here is to develop a computationally scalable and broadly applicable approach to supervised learning with invariances which is easily adapted to new types of problems and can take advantage of existing optimization infrastructures such as the bundle methods we developed in previous chapters. In this chapter we propose a method which has what we believe are many appealing properties:

1. It formulates invariant learning as a convex problem and thus can be implemented directly using any existing convex solver, requiring minimal additional memory and inheriting the convergence properties/guarantees of the underlying implementation.
2. It can deal with *arbitrary* invariances provided that the user provides a computational recipe/oracle to generate invariant equivalents efficiently from a given data point.
3. It provides a unifying framework for a number of previous approaches and is broadly applicable not just to binary classification but in fact to any structured prediction problem with a max-margin or logistic loss function.

5.2 A Unifying Formulation for Learning with Invariances

5.2.1 Invariance Transformations

The crucial ingredient to formulating invariant learning is to capture the domain knowledge that there exists some (possibly infinite) set $\mathcal{S}$ of invariance transforms $s : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$ which can act on the input $x \in \mathcal{X}$ while leaving the corresponding output $y \in \mathcal{Y}$ essentially unchanged. For instance, we might believe that slight rotation (in pixel coordinates) of an input image in a pattern recognition problem does not change the image label. For text classification problems such as email spam filtering, we may believe that certain editing operations (such as changes in capitalization or substitutions like `Viagra` $\to$ `V1agra`,`V!agra`) should not affect the "meaning" of the words in an email.

Of course, most invariances only apply "locally", *i.e.*, in the neighborhood of the original input vector. For instance, rotating an image of the digit `6` too far might change its label to `9`; applying both a substitution and an insertion can change `Viagra` $\to$ `diagram`. Furthermore, certain invariances may only hold for certain *pairs* of input and output. For example, we might believe that horizontal reflection is a valid invariance for images of digits in classes `0` and `8` but not for digits in class `2`. The set $\mathcal{S}$

incorporates both the locality and applicability constraints. To avoid degenerate cases, we assume that the set $\mathcal{S}$ includes the "identity" transform which returns the original x without change.

5.2.2 Robust Loss

With the set of invariance transforms $\mathcal{S}$, we can formulate the invariant counterpart of a standard learning problem in the form of a (constrained) robust optimization problem [Ben-Tal et al., 2009]:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \lambda\Omega(\mathbf{w}) + \frac{1}{m}\sum_{i=1}^{m}\xi_i \tag{5.1a}$$

$$\text{s.t.} \quad \forall i \in [m] : l(\hat{x}_i, y_i, \mathbf{w}) \leq \xi_i,\ \xi_i \geq 0,\ \forall \hat{x}_i \in \mathcal{S}\,|_{(x_i,y_i)}, \tag{5.1b}$$

where Ω and l are the regularizer and loss function respectively of the standard learning problem, and $\mathcal{S}\,|_{(x,y)} := \{s(x,y) \ : \ s \in \mathcal{S}\}$ is the set of all valid transformations of the input x of the example (x,y). In conventional robust optimization such as the setting of Ben-Tal et al. [2009] considered, the set $\mathcal{S}\,|_{(x,y)}$ is usually defined as an ellipsoid *e.g.*,

$$\mathcal{S}\,|_{(x,y)} := \left\{\hat{x} \in \mathbb{R}^d \ : \ (\hat{x}-x)^\top \mathbf{M}(\hat{x}-x) \leq \tau,\ x \in \mathbb{R}^d,\ \mathbf{M} \in \mathbb{R}^{d\times d}\right\}, \tag{5.2}$$

so that problem (5.1) can be solved efficiently. However, $\mathcal{S}$ can be arbitrary and depends heavily on the domain of the problem one is dealing with. For example, $\mathcal{S}$ can be a finite discrete set of transformations such as rotations of an image by p degrees for all $p \in [20]$; or replacement of all possible subsets of certain English words found in an email by their synonyms. In these cases, the constraints of (5.1) must be specified explicitly, hence, the computational complexity increases with the size of $\mathcal{S}$. If $\mathcal{S}$ is neither an ellipsoid nor finite, then (5.1) is a semi-infinite program (SIP) that may not even be readily solvable by standard SIP solvers.

We can overcome the difficulties by reformulating (5.1) into an equivalent but unconstrained problem [Teo et al., 2008]:

$$\min_{\mathbf{w}} \quad \lambda\Omega(\mathbf{w}) + R(\mathbf{w}) \tag{5.3a}$$

$$\text{where} \quad R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} l_r(x_i, y_i, \mathbf{w}), \text{ and} \tag{5.3b}$$

$$l_r(x, y, \mathbf{w}) := \max_{\hat{x}\in\mathcal{S}\,|_{(x,y)}} l(\hat{x}, y, \mathbf{w}). \tag{5.3c}$$

The equivalence between (5.1) and (5.3) can be seen by noting that (5.1b) can be

rewritten as

$$\forall i \in [m] : \max_{\hat{x}_i \in \mathcal{S}|_{(x_i, y_i)}} l(\hat{x}_i, y_i, \mathbf{w}) \leq \xi_i,\ \xi_i \geq 0,$$

and that (5.1) is in epigraph form [Boyd and Vandenberghe, 2004], hence, the slack variables ξ_i in (5.1a) can be replaced by $\max_{\hat{x}_i \in \mathcal{S}|_{(x_i, y_i)}} l(\hat{x}_i, y_i, \mathbf{w})$.

Note that l_r is necessarily convex as it is a pointwise maximum [Hiriart-Urruty and Lemaréchal, 1993] of the convex functions $l(\hat{x}, y, \mathbf{w})$, $\forall \hat{x} \in \mathcal{S}|_{(x,y)}$. Therefore, (5.3) is readily solvable by, for example, the bundle methods we developed in previous chapters. Furthermore, the set $\mathcal{S}$ need not be explicitly defined. Indeed, it can be defined implicitly as an "oracle" which, upon receiving the tuple $(x, y, \mathbf{w})$, returns the "worst" transformation $\hat{x}$ of x such that $\hat{x}$ incurs the largest loss among other possible transformations of x.

We further discuss two specific types of loss functions, namely max-margin and logistic in the following sections.

Robust Max-margin Loss

Recall the standard max-margin loss (*cf.* (2.14))

$$l(x, y, \mathbf{w}) = \max_{y' \in \mathcal{Y}} \Gamma(y, y') \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'),$$

and the predictor $y^*(x) := \operatorname{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}^*, \phi(x, y') \rangle$ with weight vector $\mathbf{w}^*$ and feature mapping $\phi(x, y) \in \mathbb{R}^d$. l is essentially a convex upper bound of the label loss Δ which enforces the penalty for incorrect prediction $\Delta(y, y^*(x))$.

For the robust counterpart, the label loss $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ can be extended to $\Delta : \mathcal{Y} \times \mathcal{Y} \times \mathcal{S} \to \mathbb{R}_+$ so that the invariance transform $s \in \mathcal{S}$, if any, which was applied to the input is taken into the account of label loss. This extension allows the label loss to depend on the transformation, for instance, we might suffer less label loss for poor predictions when the input has undergone very extreme transformations. In an image labeling problem, for example, we might believe that a lighting/exposure invariance applies but we might want to charge small label loss for extremely over-exposed or under-exposed images since they are almost impossible to label. Similarly, we might assert that scale invariance holds but give small label loss to severely spatially downsampled images since they contain very little information. Arguably, this extension takes into account the "probability" of observing such transformation which can also be regarded as another form of prior knowledge. The resulting penalty for incorrect prediction is then defined in the "worst case" sense as

$$\max_{s \in \mathcal{S}} \Delta(y, y^*(s(x, y)), s). \tag{5.4}$$

It is easy to see that (5.4) is greater than or equal to the standard label loss $\Delta(y, y^*(x))$ due to the fact that $\mathcal{S}$ includes the identity transformation.

We now define the robust max-margin loss as follows:

$$l_r(x, y, \mathbf{w}) = \max_{(y', s) \in \mathcal{Y} \times \mathcal{S}} \Gamma(y, y') \left\langle \mathbf{w}, \phi(s(x, y), y') - \phi(s(x, y), y) \right\rangle + \Delta(y, y', s). \quad (5.5)$$

This loss finds the pair of label and transformation simultaneously and penalizes the pair with incorrect label that is most compatible. The maximization over $\mathcal{Y} \times \mathcal{S}$ is the crux of the design of both the loss function and the set $\mathcal{S}$ when solving a problem: For the case where $\mathcal{Y}$ and $\mathcal{S}$ are finite and small, the maximization can be done simply via exhaustive enumeration. Otherwise, efficient algorithms are required so that the maximization is tractable. We will show in the following section that there usually exist efficient and fast algorithms for this maximization due to the linearity of the compatibility function $f(x, y) = \langle \mathbf{w}, \phi(x, y) \rangle$.

For completeness, we show in the following straightforward theorems that l_r is convex in $\mathbf{w}$ and that it upper bounds the penalty (5.4).

Lemma 5.2.1. *The loss $l_r(x, y, \mathbf{w})$ is convex in $\mathbf{w}$ for any choice of Γ, Δ and $\mathcal{S}$.*

Proof. For fixed (y', s) the expression $\Gamma(y, y') \langle \mathbf{w}, \phi(s(x, y), y') - \phi(s(x, y,), y) \rangle + \Delta(y, y', s)$ is linear in $\mathbf{w}$, hence (weakly) convex. Taking the maximum over a set of convex functions yields a convex function. □

Lemma 5.2.2. *The loss $l_r(x, y, \mathbf{w})$ provides an upper bound on $\max_{s \in \mathcal{S}} \Delta(y, y^*(s(x, y)), s)$.*

Proof. Denote by (s^*, y^*) the values for which the maximum of $\max_{s \in \mathcal{S}} \Delta(y, y^*(s(x, y)), s)$ is attained. By the definitions of $y^*(x)$ and (5.4), we have that $\langle \mathbf{w}, \phi(s^*(x, y), y^*) \rangle \geq \langle \mathbf{w}, (s^*(x, y), y) \rangle$. Plugging this inequality into l_r yields

$$l_r(x, y, \mathbf{w}) \geq \Gamma(y, y^*) \langle \mathbf{w}, \phi(s^*(x, y), y^*) - \phi(s^*(x, y), y) \rangle + \Delta(y, y^*, s^*) \geq \Delta(y, y^*, s^*).$$

Here the first inequality follows by replacing the actual maximizer of l_r by (s^*, y^*). The second inequality follows from the fact that $\Gamma \geq 0$. □

Robust Logistic Loss

Unlike the robust max-margin loss, the robust counterpart of logistic loss (*cf.* (2.13)), defined as

$$l_r(x, y, \mathbf{w}) = \max_{\hat{x} \in \mathcal{S}|_{(x,y)}} \log \left(\sum\nolimits_{y' \in \mathcal{Y}} \exp \left(\mathbf{w}^\top \phi(\hat{x}, y') \right) \right) - \mathbf{w}^\top \phi(\hat{x}, y),$$

does not involve a label loss Δ. Hence, there is no direct way to reduce the influence of extreme transformations except through the careful design of the set $\mathcal{S}$ which avoids these cases explicitly. Nevertheless, this scenario is not problematic as the set $\mathcal{S}$ is usually designed by domain experts.

5.3 Formulating Previous Approaches as Robust Loss Functions

In this section, we show that several previous approaches to invariant learning can be put under the robust learning framework proposed in this chapter.

5.3.1 Virtual Examples

The methods of virtual examples refer to the methods whereby each transform s in a *finite* set $\mathcal{S}$ of pre-defined invariance transformations is applied to some or all of the original training examples (x_i, y_i) to create "virtual" examples $(\hat{x}_i, y_i)$, $\forall \hat{x}_i \in \mathcal{S}\,|_{(x_i,y_i)}$. The main purpose of this type of invariant learning is to increase the amount and variability of training examples in the scenario where the original set of training examples is believed to be insufficient.

A notable success story of this approach of creating virtual examples for each of the original training example is the work by Loosli et al. [2007] in which a (10-class) handwritten digit classifier was trained on the well-known MNIST dataset [LeCun et al., 1995] with 134 invariance (affine) transformations. The classifier trained on the resulting 8 millions training examples achieved state-of-the-art performance on this particular dataset. However, this approach requires extra memory or computation for storing or computing the virtual examples on-the-fly.

A more memory efficient and faster alternative is the method of virtual support vectors of Schölkopf et al. [1996] (see also DeCoste and Schölkopf [2002]). Instead of creating virtual examples for *all* training examples, this method creates only virtual examples for support vectors [Schölkopf and Smola, 2002] *i.e.*, training examples which play a significant role in the decision/compatibility function. That said, this method first trains a classifier on the original set of training examples. After that, virtual examples are created for the support vectors obtained and are included in the set of training examples for the final round of training. Although the number of support vectors is known to grow linearly with the training set size, the actual number of support vectors is usually orders or magnitude smaller than the number of training examples. Hence, the final set of training examples will not be bloated by too much and training will not be as computationally intensive as that with a full set of virtual examples.

Interestingly, the methods with virtual examples created for full [*e.g.*, Loosli et al., 2007] and subset [*e.g.*, DeCoste and Schölkopf, 2002] of original training examples achieved very similar performance for handwritten digit recognition on the MNIST dataset. This observation revealed the fact that not all virtual examples are equally informative. That is, virtual examples of non-support vectors do not contribute much information. However, it is still not known in advance which training examples will end up being support vectors before the initial training.

The robust loss function addresses this issue by revealing only the most "informative" transformation (*i.e.*, the transformation which causes the largest loss) of each training example at a time (*cf.* (5.3c)), while still considering all possible transformations during the loss computation. For completeness, we note that the support vector methods of Loosli et al. [2007] and DeCoste and Schölkopf [2002] used in solving the digit recognition problem are closely related to the following robust multiclass max-margin loss:

$$l(x, y, \mathbf{W}) = \max_{s \in \mathcal{S}} \max_{y' \in [k]} (\mathbf{W}^{y'} - \mathbf{W}^{y})^{\top} s(x) + \mathbf{I}(y \neq y'),$$

where $(x, y) \in \mathbb{R}^d \times [k]$, k is the number of classes of the problem, $\mathbf{W} \in \mathbb{R}^{d \times k}$ is the weight matrix, $\mathbf{W}^i$ refers to the i-th column of $\mathbf{W}$, and $\mathbf{I}(A)$ returns 1 if the event A is true and 0 otherwise.

5.3.2 Polynomial Trajectories of Transformations

Graepel and Herbrich [2004] proposed an invariant learning approach which substitutes the input vector $x \in \mathbb{R}^d$ of a training example by a (usually infinite) set of transformed vectors

$$\mathcal{S}\,|_x := \{s(x, \theta) \; ; \; \theta \in \mathbb{R}\} \subset \mathbb{R}^d,$$

where $s : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is some transform (*e.g.*, rotation) and θ is the corresponding parameter (*e.g.*, degrees of rotation). For computational tractability, the transform s is replaced by its approximation *i.e.*, a Taylor polynomial (centered at 0) of degree r

$$\hat{s}(x, \theta) \approx \sum_{j=0}^{r} \theta^j \left(\frac{1}{j!} \frac{d^j s(x, \theta)}{d\theta^j} \Big|_{\theta=0} \right) = \sum_{j=0}^{r} \theta^j \, \mathbf{X}^j = \mathbf{X}\,\boldsymbol{\theta},$$

where $\boldsymbol{\theta} = (1, \theta, \ldots, \theta^r)$, and $\mathbf{X} \in \mathbb{R}^{d \times (r+1)}$ characterizes the polynomial trajectory corresponding to the point x. Graepel and Herbrich [2004] then formulated the hard

margin learning objective

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \tag{5.6a}$$

$$\text{s.t.} \quad \forall i \in [m] : \forall \boldsymbol{\theta} \in \mathbb{R} : y_i \, \mathbf{w}^\top \mathbf{X}_i \, \boldsymbol{\theta} \geq 1. \tag{5.6b}$$

By applying a theorem on semi-definite representations of a non-negative polynomial due to Nesterov, Graepel and Herbrich [2004] reformulated the semi-infinite program (5.6) as a semi-definite program (SDP).

We now show that (5.6) can be cast into our robust loss formulation with several desired properties. Firstly, we consider a more desirable soft margin version of constraint (5.6b):

$$\forall i \in [m] : \forall \boldsymbol{\theta} \in [p_i, q_i] : y_i \, \mathbf{w}^\top \mathbf{X}_i \, \boldsymbol{\theta} \geq 1 - \xi_i, \; \xi_i \geq 0$$

where ξ_i is a slack variable. Secondly, the constraint we consider here allows θ to take values in a segment $[p_i, q_i]$ without complicated computation as in the SDP formulation in Graepel and Herbrich [2004]. Thirdly, we rewrite the constrained problem into the robust regularized risk:

$$\min_{\mathbf{w}} \quad \lambda \Omega(\mathbf{w}) + \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, \mathbf{w}) \tag{5.7a}$$

$$\text{where} \quad l(x, y, \mathbf{w}) := \max(0, 1 - \min_{\boldsymbol{\theta} \in [p,q]} y_i \, \mathbf{w}^\top \mathbf{X} \, \boldsymbol{\theta}). \tag{5.7b}$$

The subproblem $\min_{\boldsymbol{\theta} \in [p,q]} y \, \mathbf{w}^\top \mathbf{X} \, \boldsymbol{\theta}$ can be solved by any efficient root-finding procedure for a polynomial of degree r over the interval $[p, q]$. Also, we see that (5.7) is not restricted to just the squared L_2 norm regularizer compared to the formulation of Graepel and Herbrich [2004]. Therefore, the robust loss formulation is considerably more general and scalable for high dimensional problems (*i.e.*, large d) and more accurate approximation of the invariance transformation (*i.e.*, large r).

5.3.3 Uncertainty and Missing Values in Data

In real world application of machine learning, uncertainty and missing value in data (*e.g.*, feature values of x) is one of the most common issues one has to deal with. One simple approach to this issue is by replacing/filling the uncertain/missing values with the corresponding imputed values from other data with exact values.

For training a linear classifier (with $x \in \mathbb{R}^d$), Bhattacharyya et al. [2005] (see also

Shivaswamy et al. [2006]) and Bi and Zhang [2005] proposed the robust SVM[1]:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i \tag{5.8a}$$

$$\text{s.t.} \quad \forall i \in [m] : y_i\,\mathbf{w}^\top \hat{x}_i - \gamma_i\,\|\Sigma_i\,\mathbf{w}\| \geq 1 - \xi_i,\ \xi_i \geq 0, \tag{5.8b}$$

where, in Bi and Zhang [2005], $\hat{x}_i$ is the original input vector, $\gamma_i \leq 0$, and Σ_i is an identity matrix; in Bhattacharyya et al. [2005], $\hat{x}_i$ is the mean of x_i, $\gamma_i \geq 0$, and Σ_i is the square root of the variance $\Sigma_i^2 := \Sigma_i\Sigma_i$ of x_i. Clearly, this version of SVM is an instance of the robust convex optimization formulation [Ben-Tal et al., 2009] with input vector x constrained to an ellipsoid of uncertainty (5.2) (with $\mathbf{M} := \Sigma_i$ and $\tau := \gamma_i^2$). The resulting problem is inherently a second order cone programming (SOCP) problem which is not scalable to high dimensional data.

Straightforwardly, (5.8) can be recast to the following robust regularized risk:

$$\min_{\mathbf{w}} \quad \lambda\Omega(\mathbf{w}) + \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, \mathbf{w}) \tag{5.9a}$$

$$\text{where} \quad l(x, y, \mathbf{w}) := \max(0, 1 - y_i\,\mathbf{w}^\top \hat{x}_i + \gamma_i\,\|\Sigma_i\,\mathbf{w}\|). \tag{5.9b}$$

Note that for the case where $\gamma_i < 0$, the loss (5.9b) is not convex hence the bundle methods we discussed does not guaranteed a global solution, or even worse, may not converge. Otherwise, bundle methods avoid the SOCP and hence are more efficient and scalable to large and high dimensional problems.

5.3.4 Feature Deletion as Robust Learning

Globerson and Roweis [2006] considered a setting whereby the testing environment is subject to noise or is adversary in the sense of feature corruption or deletion. To handle classification problems under this setting, Globerson and Roweis [2006] proposed a robust hinge loss for binary classification (with $x \in \mathbb{R}^d$) that models the arbitrary deletion of (up to K) features, *i.e.*, setting the feature values to zero, at training time. The robust hinge loss, FDROP, is defined as:

$$l_r(x, y, \mathbf{w}) = \max_{\mathbf{s}\in\mathcal{S}_{\{0,1\}}} \max(0, 1 - y\,\mathbf{w}^\top(x \circ \mathbf{s})) \tag{5.10}$$

where $\mathcal{S}_{\{0,1\}} = \left\{\mathbf{s} \in \{0,1\}^d \ : \ \#\{i : s_i = 1\} \geq d - K\right\}$, $\#\{\cdot\}$ denotes the cardinality of the set, and $\circ$ denotes the component-wise product. Together with the squared L_2 norm regularizer, Globerson and Roweis [2006] solves the following quadratic problem:

[1] For simplicity, we ignored the bias term of standard SVM.

$$\min_{\mathbf{w},\boldsymbol{\xi},\mathbf{t},\mathbf{z},\mathbf{v}_1,\ldots,\mathbf{v}_m} \quad \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i \tag{5.11a}$$

$$\text{s.t.} \quad \forall i \in [m] : \xi_i \geq 0,\ \xi_i \geq 1 - y_i\,\mathbf{w}^\top x_i + t_i, \tag{5.11b}$$

$$t_i \geq K z_i + \sum\nolimits_j^m v_{ij}, \tag{5.11c}$$

$$\mathbf{v}_i \geq \mathbf{0},\ z_i\mathbf{1} + \mathbf{v}_i \geq y_i(x_i \circ \mathbf{w}). \tag{5.11d}$$

$$\text{where} \quad \mathbf{w}, \mathbf{v}_i \in \mathbb{R}^d, \text{ and } \boldsymbol{\xi}, \mathbf{t}, \mathbf{z} \in \mathbb{R}^m. \tag{5.11e}$$

Clearly, problem (5.11) (or its dual) has $O(md)$ variables which renders the approach inefficient and not scalable for large and high dimensional problems. However, it is possible to devise an efficient algorithm to compute the value and subgradient of the loss function (5.10).

Note that the definition of (5.10) implies selecting a set of K features and setting their values to zero in order to maximize the loss. This is equivalent to deletion of some K features so that $y\,\mathbf{w}^\top(x \circ \mathbf{s})$ is minimized. By computing and sorting the values $u_i := yw_ix_i,\ \forall i \in [d]$ in $O(d \log d)$ time, we can find the K largest u_i for deletion; the corresponding indices identify the maximizer $\mathbf{s}$ of (5.10). Algorithm 7 provides the details of this procedure.

Algorithm 7 FDROP

1: **Input:** $x \in \mathbb{R}^d$, $y \in \{\pm 1\}$, $\mathbf{w} \in \mathbb{R}^d$, $K \geq 0$
2: Let $u_i := yw_ix_i,\ i = 1, \ldots, d$
3: Sort $\pi := [d]$ such that $u_{\pi_i} \geq u_{\pi_{i+1}},\ \forall i \in [d-1]$
4: Set $\hat{x} \leftarrow x$
5: Set $\hat{x}_{\pi_i} \leftarrow 0,\ \forall i \in [K]$
6: Compute standard hinge loss value and subgradient with $(\hat{x}, y, \mathbf{w})$

With the computed value and subgradient of the loss function, we can make use of bundle methods to solve the same problem with only $O(d)$ variables. Furthermore, the algorithm has only $O(K + d \log d)$ time complexity per loss evaluation, despite the exponentially large set $\mathcal{S}_{\{0,1\}}$ of invariance transformations.

Extensions

In fact, from FDROP, we can derive its logistic loss counterpart:

$$l_r(x, y, \mathbf{w}) = \max_{\mathbf{s} \in \mathcal{S}_{\{0,1\}}} \log(1 + \exp(-y\,\mathbf{w}^\top(x \circ \mathbf{s}))). \tag{5.12}$$

Again, due to the linearity of the term $y\,\mathbf{w}^\top(x \circ \mathbf{s})$, the value and subgradient of the robust logistic loss (5.12) can be computed by Algorithm 7 by modifying the Line 6 so that the value and gradient is computed according to standard logistic loss.

Apart from feature deletion, we can also consider a more general setting whereby the feature values are *scaled* by a factor in the range $[p, q]$, $0 \leq p \leq 1 \leq q < \infty$. This setting includes the feature deletion as a special case with $[p, q]$ replaced with $[0, 1]$ because $y\,\mathbf{w}^\top(x \circ \mathbf{s})$ is linear in $\mathbf{s}$, hence, s_i is guaranteed to take a value of either 0 or 1. The robust hinge and logistic loss functions remain the same except that the set of invariance transformations is redefined as follows:

$$\mathcal{S}_{[p,q]} := \{\mathbf{s} \in [p, q] \;:\; 0 \leq p \leq 1 \leq q < \infty,\; \#\{i : s_i = 1\} \geq d - K\}\,. \tag{5.13}$$

We refer to this robust feature scaling setting as FSCALE. Efficient algorithm akin to Algorithm 7 can be derived for computing the value and (sub)gradient of the FSCALE hinge and logistic loss functions. The efficiency of the algorithm lies in the fact that the term $y\,\mathbf{w}^\top(x \circ \mathbf{s})$ is linear in $\mathbf{s}$, and that the maximum loss can be obtained by multiplying large positive feature values with p and small negative feature values with q.

The procedure starts by computing and sorting $u_i := yw_ix_i$, $\forall i \in [d]$. Then for each of the subsequent K steps, the procedure multiplies either the largest positive u_i with p or the smallest negative u_i with q so that the loss value is increased the most. After the K scaling steps, the indices and factors corresponding to the scaled features identify the maximizer $\mathbf{s}$ of the FSCALE loss. Algorithm 8 provides the details for computing the value and (sub)gradient of FSCALE hinge or logistic loss.

Algorithm 8 FSCALE

1: **Input:** $x \in \mathbb{R}^d$, $y \in \{\pm 1\}$, $\mathbf{w} \in \mathbb{R}^d$, $K \geq 0$, $0 \leq p \leq 1 \leq q < \infty$
2: Let $u_i := yw_ix_i$, $i = 1, \ldots, d$
3: Sort $\pi := [d]$ such that $u_{\pi_i} \leq u_{\pi_{i+1}}$, $\forall i \in [d-1]$
4: $\hat{x} \leftarrow x$
5: $i \leftarrow 1$
6: $j \leftarrow d$
7: **for** $k = 1$ **to** K **do**
8: **if** $u_{\pi_i}(1-q) > u_{\pi_j}(1-p)$ **then**
9: $\hat{x}_{\pi_i} \leftarrow q\hat{x}_{\pi_i}$
10: $i \leftarrow i + 1$
11: **else**
12: $\hat{x}_{\pi_j} \leftarrow p\hat{x}_{\pi_j}$
13: $j \leftarrow j - 1$
14: **end if**
15: **end for**
16: Compute standard hinge/logistic loss value and subgradient with $(\hat{x}, y, \mathbf{w})$

5.4 Applications

In this section, we examine the use of robust loss for the setting of insufficient training examples in a handwritten digit recognition task (Section 5.4.1) and for the setting where training/test examples are corrupted by noise or adversary attacks in email spam classification (Section 5.4.2).

5.4.1 Handwritten Digit Recognition

In this experiment, we examined how useful our invariance-robust loss formulation is at improving the outcome of learning in the setting of insufficient training examples. The task we picked was training a 10-class handwritten digit recognition system on the MNIST dataset. To study the influence of insufficiency of training examples, we randomly sampled $k \in \{10, 20, \ldots, 50\}$ examples per digit from the MNIST training set (of 60,000 examples). Then, we compared the performance of three methods:

STD-SVM: standard winner-takes-all multiclass classification method [Crammer and Singer, 2003] trained on the k training examples;

VIR-SVM: STD-SVM but trained on the original k examples and the additional 20 virtual examples generated for each of the original examples (hence a total of $21k$ training examples);

Invar-SVM: STD-SVM with the original loss function replaced with the robust loss we proposed. There are only k training examples but during the loss computation, the robust loss picks the worst out of the 20 transformations and the original example.

The 20 invariance transformations we considered are: 1-pixel and 2-pixel shifts in 4 and 8 directions, rotations by ± 10 degrees, scaling by ± 0.15, and shearing in vertical or horizontal axis by ± 0.15.

All of the aforementioned SVMs were trained using the $\mathsf{SVM}^{\mathsf{multiclass}}$ implementation (an instance of $\mathsf{SVM}^{\mathsf{struct}}$ (3.13)) with RBF kernel and well-chosen hyper-parameters. For evaluation we used the standard MNIST test set (of 10,000 examples).

The experimental results for the three approaches are shown in Figure 5.1. It can be seen that Invar-SVM and VIR-SVM, which used invariances, significantly improved the recognition accuracy compared to STD-SVM for all sizes $k \in \{10, 20, \ldots, 50\}$ of the original training set; this comes at the cost of using more support vectors. The advantage of Invar-SVM is clearly reflected here as the number of support vectors is roughly half of that of the VIR-SVM, while maintaining the same performance.

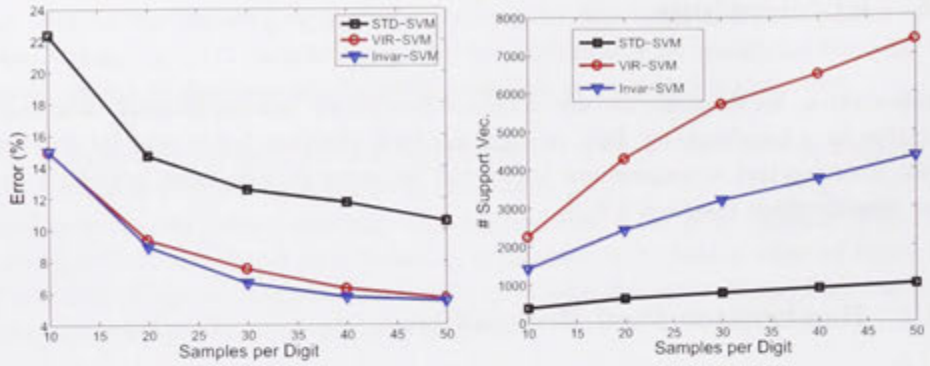


Figure 5.1: Results for the MNIST handwritten digits recognition task, comparing SVM trained on original samples (STD-SVM), SVM trained on original and virtual samples (VIR-SVM), and our invariance robust method (Invar-SVM). **Left:** Classification error on test set as a function of the number of the original training examples per digit used in training. **Right:** Number of support vectors corresponding to the optimum of each method.

5.4.2 Email Spam Classification

In real-world email spam classification, the joint distribution of both legitimate and spam emails is ever-changing in part due to the natural concept drift in the email communications (*e.g.*, change of current issues), and in part due to the adversaries *i.e.*, email spammers, who change the textual (or graphical) representation of the message in email spam in response to a newly trained classifier. For example, spammers use various tricks[2] such as the "Good Word Insertion" (GWI) and "Bad Word Obfuscation" (BWO) to make spam emails look more legitimate. GWI refers to the set of tricks which insert a random chunk of words obtained from sources such as online news websites into a spam email. Alternatively, BWO refers to the tricks which obfuscate a set words, which are likely to be blacklisted, into others which may not be found in the pre-defined dictionary; usually the obfuscated words are bogus but still recognizable by human readers.[3]

In this scenario, the representativeness of a training set and the performance of a trained classifier are very likely to *degrade* over time because the training and test sets are usually drawn from the ever-changing distribution at different time points. A straightforward approach to ameliorate this problem is by updating the training set and classifier more frequently. However, this approach will increase the computational cost as the retraining interval is decreased. An alternative is to prolong the effective life of the classifier. This is equivalent to making a classifier more robust to the increasingly more apparent difference between the training and test distributions. In other words,

[2]See "The Spammers' Compendium" at `http://www.virusbtn.com/resources/spammerscompendium/index` for a more comprehensive list of common tricks used by spammers.

[3]Some examples of word obfuscation are: `long` → `looooong`, `sex` → `se><`, `viagra` → `v1agra`, etc.

a *robust* classifier is one which degrades slower than ordinary classifiers under this setting.

In this chapter, we examine how robust loss can be useful in improving the robustness of a spam classifier trained under the setting of binary classification with spam emails being the negative class and legitimate emails being the positive class.

Experimental Setup

An email is represented by a finite d-dimensional feature vector $x \in \mathbb{R}^d$. The feature x_i can be the frequency count of the i-th word (in a pre-defined dictionary) observed in the email. It is also common to assign the feature a binary value, *i.e.*, 0 or 1, to indicate the absence or presence of that feature in the email. Under these feature representations, the difference between training and test distributions represents the fact that some words occur more/less often in the test set than in the training set. This implies that a robust classifier must not overfit or underfit itself to a small set of features. Instead, the feature weights should be spread more evenly across a larger set of features.

Since the difference between training and test distributions are not known at training time, we resort to the FSCALE loss for training a robust email spam classifier. In the experiments, we fixed $K = d$, and used the invariance transformation set

$$\mathcal{S}_{[p,q]} := \{1, u_1\} \times \cdots \times \{1, u_d\},$$

where u_i is the *importance* of the feature x_i (*e.g.*, the larger the value the more important it is). We obtained the importance vector $\mathbf{u}$ by first training a linear classifier and obtain its weight vector $\mathbf{w}$. Then u_i, $i = 1, \ldots, d$ were computed as follows:[4]

$$u_i := 1/\ln(e + |w_i|).$$

The reason behind this feature rescaling scheme is that important features found in the training set are down-weighted so that the final classifier trained is not overfitted to those features which may not be present in test examples. For a similar reason, less important features are upweighted.

Also note that, the FSCALE we considered here avoids the need of tuning the parameter K. Furthermore, the loss evaluation does not require any sorting procedure, hence, the computational complexity per loss evaluation is reduced to $O(d)$ from the original $O(K+d\log d)$. To justify the performance of FSCALE, we compared it to its non-robust counterpart with the original training examples (STD) and with training examples rescaled by the importance vector (REWEIGHT). All approaches here used the squared

[4]Alternatively, the importance vector $\mathbf{u}$ can be provided as prior knowledge or inferred from any feature ranking methods.

L_2 norm regularizer.

We performed experiments on the publicly available spam datasets: TREC 2005 [Cormack and Lynam, 2005] (TREC05), TREC 2006 English set [Cormack, 2006] (TREC06), and ECML/PKDD 2006 Discovery Challenge [Bickel, 2006] Task A evaluation set (ECML06). Details of the datasets are summarized in Table 5.1. TREC05

Dataset	d	#tr (m)	#va	#te
TREC05	208,844	24,582	6,145	61,455
TREC06	133,495	10,086	2,521	25,241
ECML06	206,908	3,200	800	7,500

Table 5.1: Details of datasets used in experiments. The second column indicates the number of features. The third through fifth columns indicate the numbers of training, validation, and test examples, respectively.

and TREC06 consist of binary-valued feature vectors and were obtained from Kołcz and Yih [2007]. ECML06 was obtained from the ECML/PKDD Discovery Challenge website[5] and the feature vectors were binarized (*i.e.*, non-zero feature values were set to 1). These datasets were further normalized to unit length in L_2 norm during the training phase. Classifiers were trained on the training set and its hyper-parameters such as the regularization parameter λ were tuned on the validation set. Final results reported here were evaluated on the test sets.

The criterion of interest in the experiments was the robustness of classifier performance subject to the GWI and BWO attacks. For the purpose of this study, we simulated at test time adversary which alternates between the two attacks [Kołcz and Teo, 2009]. At each GWI step, for each test spam input vector x, a non-existing feature (*i.e.*, $x_i = 0$) is added (*i.e.*, set $x_i = 1$) if its corresponding weight (*i.e.*, w_i) is negative and larger than that of the existing features; this means that the GWI step adds a "good" feature/word but not the "best" as spammers usually do not have access to the exact information of the classifier *i.e.*, the feature weights $\mathbf{w}$. For the BWO steps, an existing feature (*i.e.*, $x_i = 1$) of each test spam input vector x with the largest positive weight is deleted (*i.e.*, set $x_i = 0$). Here, the "worst" feature/word is assumed to be known by the spammers as they generated the spam emails. For each adversarial step, we computed the area under ROC curve (AUC) on the modified test set. Algorithm 9 provides the details of the evaluation of test performance subject to a simulated adversary.

Experimental Results

Figure 5.2 shows the results of the email spam classification experiments performed on the TREC05, TRREC06 and ECML06 datasets with STD, REWEIGHT, and FSCALE approaches using both hinge and logistic loss functions, and squared L_2 norm

[5] `http://www.ecmlpkdd2006.org/challenge.html`

Algorithm 9 Classifier performance evaluation under simulated adversarial attack

input: $\mathbf{w}$, # of attacks K, test set $D = \{(x_i, y_i)\}_{i=1}^{n} \subset \{0,1\}^d \times \{\pm 1\}$
for $k = 1$ to K **do**
  for each spam instance $(x, y) \in D$ **do**
    if k is odd **then**
      //BWO: Delete feature with largest positive weight
      Let $j = \arg\max_q \{w_q x_q\}$
      Set feature x_j to 0 if $w_j x_j > 0$
    else
      //GWI: Insert a non-existing feature which has
      // negative weight larger than the smallest
      // negative weight of any existing feature
      Let $v = \min_q \{w_q x_q\}$
      Let $j = \arg\min_q \{w_q \mid 0 > w_q \geq v \text{ and } x_q = 0\}$
      Set feature x_j to 1 if $j \neq \emptyset$
    end if
  end for
  Evaluate AUC on the L_2 norm normalized D
end for

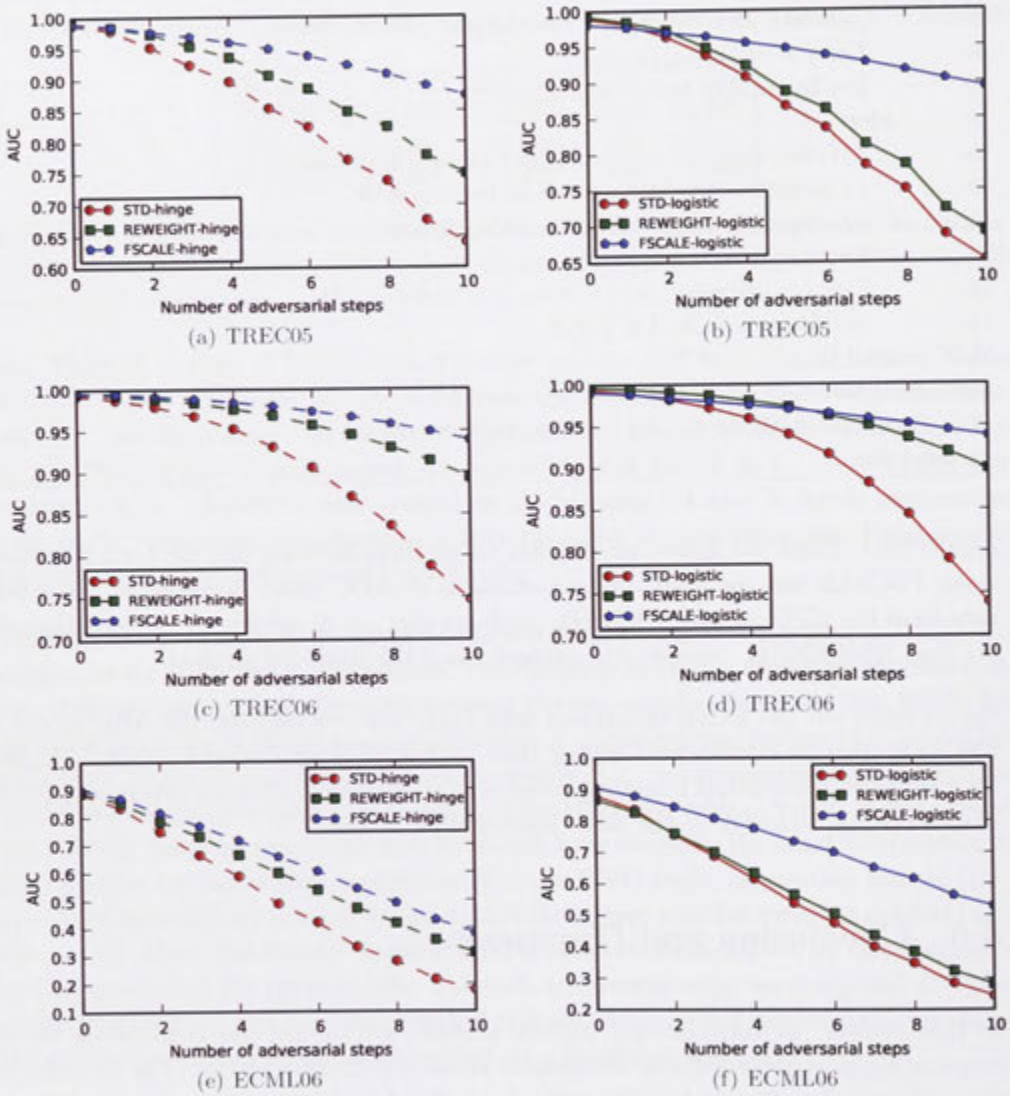
regularizers. From the figure, we see that on all three datasets and for both loss functions, FSCALE was the most robust method as its AUC scores degraded at the slower rate than the STD and REWEIGHT methods over all K adversarial steps. Also, we see that REWEIGHT consistently outperformed the standard method.

On all cases but the ECML06 dataset with hinge loss, we see that the AUC scores of FSCALE after 10 attacks were higher than that of STD after only 5 attacks. Despite its simplicity, REWEIGHT seemed to be able to withstand at least two more attacks than STD did, to arrive at the same level of AUC scores.

5.5 Conclusion and Discussion

In this chapter, we introduced a convex, general, yet robust loss formulation for incorporating prior knowledge or invariances about a problem at hand. The formulation fits well into the efficient bundle methods we developed in previous chapters, hence, many existing computationally expensive invariant learning problems can now be solved rather efficiently. Furthermore, the generality of the loss formulation opens the door for more complicated variants of invariant learning, as long as the respective loss can be evaluated exactly and efficiently. Experimental results indicated that robust losses are indeed helpful at improving the outcome of learning.

In the next chapter, we extend and develop a non-convex loss/risk for dealing with problems where the training examples are corrupted by labeling noise.



(a) TREC05 (b) TREC05

(c) TREC06 (d) TREC06

(e) ECML06 (f) ECML06

Figure 5.2: AUC evaluated on the test sets of TREC05, TREC06, and ECML06 for methods: STD, REWEIGHT, and FSCALE with hinge (left column) and logistic (right column) loss fucntions.

Chapter 6

Non-convex Loss for Structured Prediction

In this chapter, we extend a non-convex hinge loss originally developed to speed up the training of support vector machines (with non-linear kernels) to the case of structured prediction with max-margin loss. We show in numerical experiments that the non-convex loss is more robust to labeling noise (*i.e.*, the noise present in the labeling process) than the convex counterpart.

6.1 Noisy Data Meet Convex Loss

As described in Section 2.1, the risk (*i.e.*, the sum of *actual* losses computed on training examples) represent a measure of goodness of a predictor. Due to the discontinuity of the actual loss in the parameter of interest, that is, the weight vector $\mathbf{w}$, the loss is substituted with a convex surrogate such as the max-margin loss (2.14) or the logistic loss (2.13) which is continuous in $\mathbf{w}$.

The convexity of the surrogate loss functions helps to reduce intractable learning problems into ones that are readily solvable by well-studied and efficient numerical optimization techniques. However, convex loss functions have an undesirable property that the loss value grows without bound. For example, the hinge loss for binary classification

$$l(x, y, \mathbf{w}) = \max(0, 1 - y \langle \mathbf{w}, x \rangle) \tag{6.1}$$

scales linearly when an example is not very well classified, that is, the value $y \langle \mathbf{w}, x \rangle$ is less than 1. This implies that the loss values of those noisy or mislabeled examples can dominate and ruin the empirical risk. As a result, the risk minimization will be misled and the solution obtained is likely to be suboptimal.

We illustrate in Figure 6.1 a toy binary classification problem solved with linear SVMs. From the figure, we see that mislabeled points distort the decision function (middle

plot), and are otherwise not affecting the original (left plot) when removed (right plot).

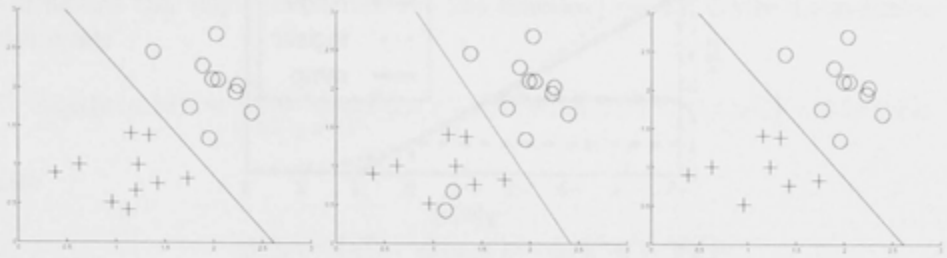


Figure 6.1: Outcomes of SVM training on original data set (left plot), on the same data set but with two points mislabeled (middle plot), and on the same data set with mislabeled points removed (right plot). Positive points are marked with plus signs '+' and negative points with circles 'o'. The solid line is the SVM decision boundary.

The mislabeling problem is a common challenge in machine learning. This problem is more severe in the case of structured prediction as there are frequently many labels that match an input equally well but only one of them is deemed to be correct. In the following section we see how this problem is handled with a non-convex loss.

6.2 A Non-convex Loss

In binary classification, Collobert et al. [2006] proposed to switch from the convex hinge loss (6.1) to a tighter non-convex loss, namely the ramp loss

$$l_{\text{ramp}}(x, y, \mathbf{w}) := \min(1 + \kappa, \max(0, 1 - y \langle \mathbf{w}, x \rangle)) \tag{6.2a}$$

$$= \max(0, 1 - y \langle \mathbf{w}, x \rangle) - \max(0, -\kappa - y \langle \mathbf{w}, x \rangle), \tag{6.2b}$$

where $\kappa \geq 0$. Their original aim of using the ramp loss was not for obtaining better classification accuracy but for speeding up the training of SVMs due to the decreased number of support vectors. Figure 6.2 illustrates the ramp loss along with 0/1, hinge, and logistic losses. Clearly from the figure, we see that convex surrogate losses scale at least linearly with the negative value of $y \langle \mathbf{w}, x \rangle$ which deviates greatly from the constant value of the original 0/1 loss. The ramp loss alleviates the over-penalization by truncating the loss value at a user-specified threshold (*e.g.*, κ in (6.2)), hence the loss value does not increase unboundedly.

A closer look at (6.2) reveals that l_{ramp} is in fact a difference of two convex functions. This observation allows Collobert et al. [2006] to use the concave-convex procedure (CCCP) of Yuille and Rangarajan [2003], which is also well known in optimization as the difference of convex (DC) programming method [Tuy, 1995]. We extend the notion of ramp loss beyond the scope binary classification. Furthermore, we show that the

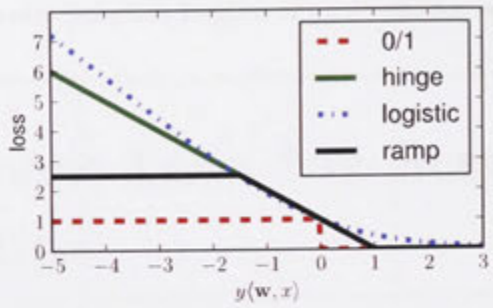


Figure 6.2: 0/1 loss, convex surrogates: hinge loss and logistic loss, and non-convex surrogate: ramp loss.

algorithm used for solving the risk minimization with ramp loss [Collobert et al., 2006] is easily amenable to the case of structured prediction.

6.2.1 Non-convex Max-margin Loss

We redefine the max-margin loss (2.14) as

$$l(x, y, \mathbf{w}) := \max_{y' \in \mathcal{Y}} \Gamma(\Delta(y, y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'). \tag{6.3}$$

where $\Gamma : [0, \infty) \to [0, \infty)$ is monotonically non-decreasing. Despite the redefinition of Γ, (6.3) still generalizes the formulation of Tsochantaridis et al. [2005] with $\Gamma := \Delta$ and that of Taskar et al. [2004] with $\Gamma := 1$. We further state a generalized version of (6.3) as the following:

$$l(x, y, y', \mathbf{w}) := \max_{y'' \in \mathcal{Y}} \Gamma(\Delta(y, y'')) \left\langle \mathbf{w}, \phi(x, y'') - \phi(x, y') \right\rangle + \Delta(y, y''). \tag{6.4}$$

The following theorem establishes the convexity of (6.4).

Theorem 6.2.1. *l defined in (6.4) is convex in $\mathbf{w}$. Furthermore,*

$$l(x, y, y', \mathbf{w}) \geq \Delta\left(y, y^*(x, \mathbf{w})\right)$$

for all $y, y' \in \mathcal{Y}$, where $y^*(x, \mathbf{w}) := \operatorname{argmax}_{y'' \in \mathcal{Y}} \langle \mathbf{w}, \phi(x, y'') \rangle$.

Proof. Convexity follows immediately from the fact that l is the maximum over linear functions in $\mathbf{w}$. To see the inequality, substitute the maximizer of (6.4) by $y'' = y^*(x, \mathbf{w})$ and use the fact that $\langle \mathbf{w}, \phi(x, y^*(x, \mathbf{w})) \rangle \geq \langle \mathbf{w}, \phi(x, y'') \rangle$ for all $y'' \in \mathcal{Y}$. □

Chapelle et al. [2007] observed that there are cases where more than one optimal (*i.e.*, most compatible) label is suitable for an example in structured prediction, especially,

in document ranking where the labels are permutations. In these cases, a predicted label which is different but *equivalent* to the optimal one should not incur (much) loss. To formulate this requirement into the loss function, we can devise a non-convex loss which reads

$$l_{\text{tight}}(x, y, \mathbf{w}) := \min_{y'' \in \mathcal{Y}|_y} \max_{y' \in \mathcal{Y}} \Gamma(\Delta(y'', y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y'') \right\rangle + \Delta(y'', y'),$$

where

$$\mathcal{Y}|_y := \left\{ y'' \ : \ \Delta(y, y') = \Delta(y'', y'), \ \forall y' \right\}$$

is a set of imposter labels *equivalent* to the genuine label y of input x. Clearly, l_{tight} is upper bounded by l in (6.3) by the definition of (6.4). Essentially, l_{tight} replaces the actual label y with an equivalent one from the set $\mathcal{Y}|_y$ which incurs the smallest loss. In other words, this modification prevents the risk minimization procedure from enforcing the margin separation between the genuine and the imposter input-label pairs:

$$\langle \mathbf{w}, \phi(x, y) \rangle \geq \left\langle \mathbf{w}, \phi(x, y') \right\rangle, \ \forall y' \in \mathcal{Y} \backslash y,$$

when there are ambiguous labels, or when y is an incorrect label.

Unfortunately, in structured prediction, generating the equivalence set $\mathcal{Y}|_y$ is intractable when $\mathcal{Y}$ is exponentially large. Hence, Chapelle et al. [2007, 2009] proposed to use the following non-convex loss:

$$\begin{aligned} \bar{\bar{l}}(x, y, \mathbf{w}) := & \max_{y' \in \mathcal{Y}} \Gamma(\Delta(y, y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y') \\ & - \max_{y'' \in \mathcal{Y}} \Gamma(\Delta(y, y'')) \left\langle \mathbf{w}, \phi(x, y'') - \phi(x, y) \right\rangle . \end{aligned} \tag{6.5}$$

It is easy to see that for $(x, y) \in \mathbb{R}^d \times \{\pm 1\}$, $\Gamma(\eta) = 1$, $\phi(x, y) = \frac{1}{2} yx$, and $\Delta(y, y') = \frac{1}{2}|y - y'|$, we recover the ramp loss (6.2). Also, it is obvious that $\bar{\bar{l}}$ is upper bounded by the standard max-margin loss l since the second term of $\bar{\bar{l}}$ (not including the sign) cannot attain a negative value.

We now show that $\bar{\bar{l}}$ is a (continuous) tighter upper bound of the label loss Δ.

Theorem 6.2.2. *Denote as follow the predictor, margin violator, and rescaled estimate*

$$\begin{aligned} y^* &:= \operatorname*{argmax}_{y'} \left\langle \mathbf{w}, \phi(x, y') \right\rangle, \\ y^{\text{m}} &:= \operatorname*{argmax}_{y'} \Gamma(\Delta(y, y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle, \\ y^{\text{l}} &:= \operatorname*{argmax}_{y'} \Gamma(\Delta(y, y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'). \end{aligned} \tag{6.6}$$

Let $\bar{\bar{l}}$ *be as defined in (6.5). The following inequalities hold for all* $\mathbf{w}$*:*

$$\Delta(y, y^{\mathrm{l}}) \geq \bar{\bar{l}}(x, y, \mathbf{w}) \geq \Delta(y, y^{\mathrm{m}}) \geq \Delta(y, y^*). \tag{6.7}$$

Proof. Assume any $\mathbf{w}$. Since y^{l} maximizes the first term of $\bar{\bar{l}}$, replacing the maximizer y' of (6.5) by y^{l} yields

$$\begin{aligned}\bar{\bar{l}}(x, y, \mathbf{w}) &\leq \Gamma(\Delta(y, y^{\mathrm{l}}))\left\langle \mathbf{w}, \phi(x, y^{\mathrm{l}}) - \phi(x, y)\right\rangle + \Delta(y, y^{\mathrm{l}}) \\ &\quad - \Gamma(\Delta(y, y^{\mathrm{l}}))\left\langle \mathbf{w}, \phi(x, y^{\mathrm{l}}) - \phi(x, y)\right\rangle \\ &= \Delta(y, y^{\mathrm{l}}),\end{aligned}$$

which proves the first inequality. Similarly, the second inequality $\bar{\bar{l}}(x, y, \mathbf{w}) \geq \Delta(y, y^{\mathrm{m}})$ is proved by replacing the maximizer y' of (6.5) by y^{m}. We now prove the third inequality $\Delta(y, y^{\mathrm{m}}) \geq \Delta(y, y^*)$. For this purpose, we distinguish the following two cases:

Case 1: $y^* = y^{\mathrm{m}}$.
The third inequality turns into equality trivially.

Case 2: $y^* \neq y^{\mathrm{m}}$.
By the definitions of y^* and y^{m}, $\langle \mathbf{w}, \phi(x, y^*)\rangle \geq \langle \mathbf{w}, \phi(x, y^{\mathrm{m}})\rangle$, hence, we have that $\Gamma(\Delta(y, y^{\mathrm{m}}))\,\langle \mathbf{w}, \phi(x, y^*) - \phi(x, y)\rangle \geq \Gamma(\Delta(y, y^*))\,\langle \mathbf{w}, \phi(x, y^*) - \phi(x, y)\rangle$ and thus $\Gamma(\Delta(y, y^{\mathrm{m}})) > \Gamma(\Delta(y, y^*))$. Since Γ is non-decreasing this implies $\Delta(y, y^{\mathrm{m}}) > \Delta(y, y^*)$.

Therefore, we have completed the proof. □

Note that the main difference between the case of constant Γ and monotonic Γ is that in the latter case the bounds are not quite as tight as they could potentially be, since we still have some slack with respect to $\Delta(y, y^{\mathrm{m}})$. Monotonic Γ tends to overscale the margin such that more emphasis is placed on avoiding large deviations from the correct estimate rather than restricting small deviations.

6.2.2 DC Programming for Non-convex Max-margin Loss

We briefly review the basic template of DC programming, as described in Yuille and Rangarajan [2003]. For a function $h : \mathbb{R}^d \to \mathbb{R}$, defined by

$$h(\mathbf{w}) = h_{\cup}(\mathbf{w}) + h_{\cap}(\mathbf{w}),$$

which can be expressed as the sum of a convex $h_\cup$ and a concave $h_\cap$ function, we can find a convex upper bound by

$$h_\cup(\mathbf{w}) + h_\cap(\mathbf{w}_0) + \left\langle \mathbf{w} - \mathbf{w}_0, \mathbf{g}_{\mathbf{w}_0} \right\rangle, \tag{6.8}$$

where $\mathbf{g}_{\mathbf{w}_0} \in \partial h_\cap(\mathbf{w}_0)$. This follows from the linearization (*i.e.*, first-order Taylor expansion) of the concave term $h_\cap$ at the current value of $\mathbf{w}_0$. Subsequently, the upper bound (6.8) is minimized instead of the function h. After that, a new linearization of $h_\cap$ is computed, and the procedure is repeated until convergence. This will lead to a local minimum of h, as shown in Yuille and Rangarajan [2003].

We now proceed to deriving an explicit instantiation of DC programming for structured prediction with the non-convex max-margin loss $\bar{\bar{l}}$ in (6.5). We first provide the linearization for the concave term of $\bar{\bar{l}}$ as the following:

$$-\Gamma(\Delta(y, y^{\mathrm{m}})) \left\langle \mathbf{w}, \phi(x, y^{\mathrm{m}}) - \phi(x, y) \right\rangle \geq - \max_{y'} \Gamma(\Delta(y, y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle,$$

where y^{m} is a margin violator as defined in (6.6). This leads to the following convex upper bound of $\bar{\bar{l}}$:

$$\begin{aligned} \tilde{l}(x, y, y^{\mathrm{m}}, \mathbf{w}) := & \max_{y' \in \mathcal{Y}} \Gamma(\Delta(y, y')) \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y') \\ & - \Gamma(\Delta(y, y^{\mathrm{m}})) \left\langle \mathbf{w}, \phi(x, y^{\mathrm{m}}) - \phi(x, y) \right\rangle. \end{aligned} \tag{6.9}$$

In the case of $\Gamma(\eta) = 1$ this can be simplified significantly: the terms in $\langle \mathbf{w}, \phi(x, y) \rangle$ cancel and $\tilde{l}$ becomes

$$\tilde{l}(x, y, y^{\mathrm{m}}, \mathbf{w}) = \max_{y' \in \mathcal{Y}} \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y^{\mathrm{m}}) \right\rangle + \Delta(y, y'). \tag{6.10}$$

In other words, we replace the correct label y by the margin violator y^{m}. Such modifications can be easily implemented in the bundle method solvers and related algorithms which only require access to the gradient information (and the function value). In fact, the above strategy follows directly from Theorem 6.2.1 when replacing y' by the margin violator y^{m}. Algorithm 10 lists the details for solving the regularized risk corresponding to the non-convex max-margin loss (6.5) using the CCCP algorithm.

6.3 Experiment: Multiclass Classification with Noisy Data

To illustrate the use of non-convex max-margin loss, we performed multiclass classification experiments on several standard datasets with artificial labeling noise of different magnitudes introduced. We compared the performance of the convex and non-convex max-margin loss for multiclass classification.

Algorithm 10 Structured Prediction with Non-convex Max-margin Loss

input: termination tolerance $\delta > 0$
Let $R(\mathbf{w}) := \sum_{i=1}^{m} \tilde{l}(x_i, y_i, y_i, \mathbf{w})$
Solve $\mathbf{w}_1 \leftarrow \operatorname{argmax}_{\mathbf{w}} J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w})$ using, for example, BMRM
$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  Compute $y_i^{\mathrm{m}} := \operatorname{argmax}_{y'} \Gamma(\Delta(y_i, y')\, \langle \mathbf{w}_t, \phi(x_i, y') \rangle)$ for all $i \in [m]$
  Let $R(\mathbf{w}) := \sum_{i=1}^{m} \tilde{l}(x_i, y_i, y_i^{\mathrm{m}}, \mathbf{w})$
  Solve $\mathbf{w}_{t+1} \leftarrow \operatorname{argmax}_{\mathbf{w}} J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R(\mathbf{w})$
until $J(\mathbf{w}_{t+1}) - J(\mathbf{w}_t) \leq \delta$

The convex loss is the specialization of (6.3) where $\mathcal{Y} = [c]$ with c the number of classes, $\mathcal{X} = \mathbb{R}^d$, $\phi(x, y) = e_y \otimes x \in \mathbb{R}^{dc}$ with e_i a vector of all zeros except the y-th entry being 1 and $\otimes$ denotes Kronecker product, $\Gamma(\eta) = 1$, and $\Delta(y, y') = \mathbf{I}(y \neq y')$. More clearly, the convex loss is

$$l(x, y, \mathbf{w}) := \max_{y' \in [c]} \left\langle \mathbf{w}, e_{y'} \otimes x - e_y \otimes x \right\rangle + \mathbf{I}(y \neq y'), \tag{6.11}$$

which has previously been introduced in (3.10). The non-convex loss corresponding to the convex loss (6.11) reads

$$l(x, y, \mathbf{w}) := \max_{y' \in [c]} \left\langle \mathbf{w}, e_{y'} \otimes x \right\rangle + \mathbf{I}(y \neq y') - \max_{y'' \in [c]} \left\langle \mathbf{w}, e_{y''} \otimes x \right\rangle.$$

For both loss functions, we formed the corresponding regularized risks with squared L_2 norm regularizer.

The UCI/Statlog datasets used in the experiments were: dna, letter, satimage, segment, shuttle, and usps. These datasets were downloaded from the LIBSVM tools website.[1] To keep the proportion of labels fixed, we introduced artificial labeling noise by randomly swapping $p \in \{10, 20\}$ percents of the labels in the training set. Also, the labeling noise was introduced in a stratified fashion, that is, we chose a fixed fraction of examples from each of the c classes and permuted their label assignments randomly. We also compared the convex and the non-convex loss on the original datasets without artificial labeling noise.

Table 6.2 shows the results in average accuracy $\pm$ standard deviation on several datasets with different percentages of labels shuffled. We used nested 10-fold cross validation to adjust the regularization parameter λ and to compute the accuracy. It can be seen that the non-convex loss outperformed the convex loss on almost all datasets when the datasets were noisy (*i.e.*, for the cases of 10% and 20% labeling noise). On the original "clean" dataset, the convex loss was slightly superior. The results supported

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

Dataset	# examples m	# classes c	dimension d
dna	3,186	3	180
letter	20,000	26	16
satimage	6,435	6	36
segment	2,310	7	19
shuttle	58,000	7	9
usps	9,298	10	256

Table 6.1: Properties of multiclass classification datasets used in the experiments.

our conjecture that, tighter non-convex losses are more robust to data with labeling noise, compared to convex losses.

Dataset	Methods	0%	10%	20%
dna	convex	95.2 ± 1.1	88.9 ± 1.5	83.1 ± 2.4
	non-convex	95.1 ± 0.8	89.1 ± 1.3	83.5 ± 2.2
letter	convex	76.8 ± 0.9	64.6 ± 0.7	50.1 ± 1.4
	non-convex	78.6 ± 0.8	70.8 ± 0.8	63.0 ± 1.5
satimage	convex	85.1 ± 0.9	77.0 ± 1.6	66.4 ± 1.3
	non-convex	85.4 ± 1.2	78.1 ± 1.6	70.7 ± 1.0
segment	convex	95.4 ± 0.9	84.8 ± 2.3	73.8 ± 2.1
	non-convex	95.2 ± 1.0	85.9 ± 2.1	77.5 ± 2.0
shuttle	convex	97.4 ± 0.2	89.5 ± 0.2	83.8 ± 0.2
	non-convex	97.1 ± 0.2	90.6 ± 0.8	88.1 ± 0.3
usps	convex	95.1 ± 0.7	85.3 ± 1.3	76.5 ± 1.4
	non-convex	95.1 ± 0.9	86.1 ± 1.6	77.6 ± 1.1

Table 6.2: Average accuracy and standard deviation for multiclass classification using the convex and the non-convex max-margin loss functions. The third through fifth columns represent results for datasets with none, 10%, and 20% of the labels randomly shuffled, respectively.

6.4 Summary and Related Works

We proposed a simple modification of the convex max-margin loss used in structured prediction which can be used to obtain tighter (albeit non-convex) bounds on sophisticated label loss functions. The advantage of the non-convex loss is that it requires next to no modification of existing optimization algorithms but rather repeated invocation of a structured prediction solver such as BMRM. The experimental results showed that non-convex loss is more robust to noisy data than convex loss.

Interestingly, a non-convex max-margin loss similar to (6.5) has also been used for learning with latent variables by Yu and Joachims [2009]. Briefly, the loss considered

by Yu and Joachims [2009] is

$$l(x, y, \mathbf{w}) := \min_{z \in \mathcal{Z}} \max_{(y', z') \in \mathcal{Y} \times \mathcal{Z}} \left\langle \mathbf{w}, \phi(x, y', z') - \phi(x, y, z) \right\rangle + \Delta((y, z), (y' z'))$$
$$= \max_{(y', z') \in \mathcal{Y} \times \mathcal{Z}} \left\langle \mathbf{w}, \phi(x, y', z') \right\rangle + \Delta((y, z), (y' z')) - \max_{z \in \mathcal{Z}} \left\langle \mathbf{w}, \phi(x, y, z) \right\rangle \quad (6.12)$$

where $\mathcal{Z}$ is a space of latent variables z, $\phi : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}^d$ and $\Delta : (\mathcal{Y} \times \mathcal{Z})^2 \to \mathbb{R}_+$ are the extensions of normal ϕ and Δ, respectively, to incorporate the latent variable z. If we think of the original training labels are incomplete, that is, in the sense of Yu and Joachims [2009], without the latent variables, then (6.12) can be linked to l_{tight} with $\mathcal{Y} := \mathcal{Y} \times \mathcal{Z}$, $\mathcal{Y}|y := \{(y, z) \ : \ (y, z'), \ \forall z' \in \mathcal{Z}\}$ and $\Gamma = 1$.

Along the line of learning with latent/missing variables, Smola et al. [2005] proposed a more general framework of estimation problems in exponential families. The resulting optimization problems are non-convex but can be solved efficiently by DC programming.

Chapter 7

Conclusion

We conclude this thesis with a summary of our contributions and a discussion on future work.

7.1 Contributions

Below we list and detail the contributions of this thesis on developing an optimization method for solving learning problems, and two risk formulations for improving the performance and robustness of the learned model.

- **Faster bundle method for regularized risk minimization**
  We developed a bundle method specialized to regularized risk minimization. We showed that the convergence rate of the method is of order $O(\epsilon^{-1})$ for strongly convex regularizers with any convex regularized risk. Furthermore, the convergence rate is enhanced to $O(\log(\epsilon^{-1}))$ when the empirical risk is differentiable. These rates are better than the $O(\epsilon^{-1}\log(\epsilon^{-1}))$ rate we sketched for the proximal bundle method *i.e.*, Theorem 2.3.1 in Section 2.3.3.

- **Efficient computation of approximate regularization path and model selection**
  The bundle method we developed produces an approximation (*i.e.*, a piecewise linear lower bound) only for the empirical risk instead of the whole regularized risk, compared to standard bundle methods. This leads to efficient regularization path computation and model selection because changes in regularization parameters λ will not invalidate the approximation of the empirical risk. By retaining the approximation built in the previous optimization round (*i.e.*, with a larger λ), the new optimization round (*i.e.*, with a smaller λ) can be greatly sped up. Empirical results showed that the reuse of approximations significantly improves the convergence compared to standard warm-start strategies where only a good starting point is provided (but not the approximation). We note that there exist exact regularization path algorithms for convex piecewise linear risks with a

reasonable number of hinges, and approximate algorithms for smooth risks. Our work fills the gap for structured prediction problems where the max-margin risks have possibly exponentially many hinges.

- **Modular and scalable implementation of bundle method for regularized risk minimization**
  Due to the modular nature of regularized risk minimization, many different regularized risks can be implemented quickly by a simple mix-and-match approach based on a set of implementations of regularizers and empirical risks. Moreover, for decomposable empirical risks, the computation can be fully parallelized and distributed over multiple cores/machines to speed up computation. In particular, this parallel and distributed bundle method provides an efficient and feasible alternative for collaborated learning with privacy sensitive and geographically distributed data.

- **Convex robust risk for incorporation of invariance and prior knowledge**
  The convex worst-case risk formulation unifies many seemingly different approaches for learning in environments with irregularities. These irregularities include missing or uncertain feature values and covariate shift. The latter, in turn, includes special cases such as insufficient training data and adversarial test environments. In particular, we showed a worst-case risk targeted at classification with missing features at test time can be solved more efficiently by the bundle methods (and other gradient based methods) than by the originally proposed quadratic programming formulation.

- **Robust classifier training for email spam classification**
  We used the convex worst-case risk for training email spam classifiers with improved robustness to adversaries in the test environment. In addition, the procedure for simulating the adversaries is novel and provides a lower bound on the damage that could possibly caused by real adversary. The same adversary simulation procedure can be used in combination with the area under ROC as a criteria in model selection.

- **Extending non-convex loss for structured prediction**
  We extended a non-convex risk formulation, originally proposed for binary classification with outliers, to the case of structured prediction and demonstrated its usefulness in experiments.

7.2 Future Work

We list some possible future work in this section.

- **Fully parallelized and distributed bundle methods**
  Algorithm 4 is not fully parallelized and distributed as the subproblem for updat-

ing the new iterate is done on the master node. One simple but tedious avenue for future work would be to parallelize the subproblem computation and to decentralize the storage of linearizations. The benefit of doing this is twofold. Firstly, by parallelization, the subproblem is likely to be solved faster than by Algorithm 4. Secondly, distributing the linearizations over different machines allows the bundle methods to keep more linearizations (in memory) when the dimensionality of the problem is high and hence produce a more accurate approximation of the empirical risk. As was shown in the experiments in Section 3.4.1, a larger bundle of linearizations always leads to faster convergence. To this end, publicly available numerical libraries for parallel and distributed linear algebra and optimization such as PETSc [Balay et al., 1997] and TAO [Benson et al., 2007] can be used.

- BMRM **with adaptive regularization**
  When a regularizer is strongly convex, the regularization parameter λ serves as a multiplicative factor to the modulus of strong convexity of the regularized risk. As shown in Theorem 3.2.2, the convergence is inversely proportional to the regularization parameter (*i.e.*, $O(\lambda^{-1})$). Therefore, the larger the value of λ, the faster BMRM converges. For a user-specified λ, we can follow the numerical continuation methods [Allgower and Georg, 2003] to replace the original optimization by a sequence of similar optimizations with λ and ϵ (*i.e.*, the desired optimization accuracy) changes (maybe non-monotonically) from a large value to their smaller original values. It is likely that solving this sequence of optimizations is faster than solving the original optimization with small λ and ϵ.

- **Generalization ability of (non)-convex robust risks**
  Although the convex and non-convex risk formulations were shown to improve learning performance in various irregular training/test environments, the conventional generalization analyses do not seem to apply fully to them. Further theoretical analysis is needed to fortify the understanding of these risk formulations. Related work can be found in Candela et al. [2009] and references therein.

Appendix A

Loss Functions

We give an expository of many commonly used convex differentiable and non-differentiable loss functions; Tables A.1 contains a choice subset of such loss functions. We further discuss several more *sophisticated* loss/risk functions in the subsequent sections.

A.1 Loss Functions for Structured Prediction

In recent years structured prediction has gained substantial popularity in machine learning [Tsochantaridis et al., 2005, Taskar et al., 2004, Bakir et al., 2007]. At its core it relies on two types of convex loss functions: logistic loss:

$$l(x, y, \mathbf{w}) = \log \sum_{y' \in \mathcal{Y}} \exp\left(\langle \mathbf{w}, \phi(x, y') \rangle\right) - \langle \mathbf{w}, \phi(x, y) \rangle \,, \tag{A.1}$$

and max-margin loss:

$$l(x, y, \mathbf{w}) = \max_{y' \in \mathcal{Y}} \Gamma(y, y') \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'). \tag{A.2}$$

Here $\phi(x, y)$ is a feature map, $\Delta(y, y') \geq 0$ describes the cost of misclassifying y by y', and $\Gamma(y, y') \geq 0$ is a scaling term which indicates by how much the large margin property should be enforced. For instance, Taskar et al. [2004] choose $\Gamma(y, y') = 1$. On the other hand Tsochantaridis et al. [2005] suggest $\Gamma(y, y') = \Delta(y, y')$, which reportedly yields better performance. Finally, McAllester [2007] recently suggested generic functions $\Gamma(y, y')$.

The logistic loss can also be interpreted as the negative log-likelihood of a conditional exponential family model:

$$p(y|x; \mathbf{w}) := \exp(\langle \mathbf{w}, \phi(x, y) \rangle - g(\mathbf{w}\,|x)), \tag{A.3}$$

	Loss $l(x, y, \mathbf{w})$	Derivative $\partial_{\mathbf{w}} l(x, y, \mathbf{w})$
Hinge [Bennett and Mangasarian, 1992]	$\max(0, 1 - yf)$	0 if $yf \geq 1$ and $-yx$ otherwise
Squared Hinge [Keerthi and DeCoste, 2005]	$\frac{1}{2}\max(0, 1 - yf)^2$	0 if $yf \geq 1$ and $(f - y)x$ otherwise
Exponential [Cowell et al., 1999]	$\exp(-yf)$	$-y\exp(-yf)x$
Logistic [Collins et al., 2000]	$\log(1 + \exp(-yf))$	$-y/(1 + \exp(-yf))x$
Novelty [Schölkopf et al., 2001]	$\max(0, \rho - f)$	0 if $f \geq \rho$ and $-1x$ otherwise
Least mean squares [Williams, 1998]	$\frac{1}{2}(f - y)^2$	$(f - y)x$
Least absolute deviation	$\lvert f - y \rvert$	$\text{sign}(f - y)x$
Quantile regression [Koenker, 2005]	$\max(\tau(f - y), (1 - \tau)(y - f))$	τx if $f > y$ and $(\tau - 1)x$ otherwise
ϵ-insensitive [Vapnik et al., 1997]	$\max(0, \lvert f - y \rvert - \epsilon)$	0 if $\lvert f - y \rvert \leq \epsilon$, else $\text{sign}(f - y)x$
Huber's robust loss [Müller et al., 1997]	$\frac{1}{2}(f - y)^2$ if $\lvert f - y \rvert \leq 1$, else $\lvert f - y \rvert - \frac{1}{2}$	$(f - y)x$ if $\lvert f - y \rvert \leq 1$, else $\text{sign}(f - y)x$
Poisson regression [Cressie, 1993]	$\exp(f) - yf$	$(\exp(f) - y)x$

Table A.1: Loss functions and their derivatives. We denote $f := \langle \mathbf{w}, x \rangle$.

where the normalizing constant $g(\mathbf{w}\,|x)$, often called the log-partition function, reads

$$g(\mathbf{w}\,|x) := \log \sum_{y' \in \mathcal{Y}} \exp\left(\langle \mathbf{w}, \phi(x, y') \rangle\right). \tag{A.4}$$

As a consequence of the Hammersley-Clifford theorem [Jordan, 2002] every exponential family distribution corresponds to a undirected graphical model. In our case this implies that the labels y factorize according to an undirected graphical model. A large number of problems have been addressed by this setting, amongst them named entity tagging [Lafferty et al., 2001], sequence alignment [Tsochantaridis et al., 2005], segmentation [Rätsch et al., 2007] and path planning [Ratliff et al., 2006]. It is clearly impossible to give examples of all settings in this section, nor would a brief summary do this field any justice. We therefore refer the reader to the edited volume Bakir et al. [2007] and the references therein.

If the underlying graphical model is tractable then efficient inference algorithms based on dynamic programming can be used to compute (A.1) and (A.2). We discuss intractable graphical models in Section A.1.1, and now turn our attention to the derivatives of the above structured losses.

When it comes to computing derivatives of the logistic loss, (A.1), we have

$$\partial_{\mathbf{w}} l(x, y, \mathbf{w}) = \frac{\sum_{y'} \phi(x, y') \exp \langle \mathbf{w}, \phi(x, y') \rangle}{\sum_{y'} \exp \langle \mathbf{w}, \phi(x, y') \rangle} - \phi(x, y) \tag{A.5}$$

$$= \mathbf{E}_{y' \sim p(y'|x)} \left[\phi(x, y')\right] - \phi(x, y). \tag{A.6}$$

where $p(y|x)$ is the exponential family model (A.3). In the case of (A.2) we denote by $\bar{y}(x)$ the argmax of the RHS, that is

$$\bar{y}(x) := \operatorname*{argmax}_{y'} \Gamma(y, y') \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'). \tag{A.7}$$

This allows us to compute the derivative of $l(x, y, \mathbf{w})$ as

$$\partial_{\mathbf{w}} l(x, y, \mathbf{w}) = \Gamma(y, \bar{y}(x)) \left[\phi(x, \bar{y}(x)) - \phi(x, y)\right]. \tag{A.8}$$

In the case where the loss is maximized for more than one distinct value $\bar{y}(x)$ we may average over the individual values, since any convex combination of such terms lies in the subdifferential.

Note that (A.2) majorizes $\Delta(y, y^*)$, where $y^* := \operatorname{argmax}_{y'} \langle \mathbf{w}, \phi(x, y') \rangle$ [Tsochantaridis et al., 2005]. This can be seen via the following series of inequalities:

$$\Delta(y, y^*) \leq \Gamma(y, y^*) \langle \mathbf{w}, \phi(x, y^*) - \phi(x, y) \rangle + \Delta(y, y^*) \leq l(x, y, \mathbf{w}).$$

The first inequality follows because $\Gamma(y, y^*) \geq 0$ and y^* maximizes $\langle \mathbf{w}, \phi(x, y') \rangle$ thus implying that $\Gamma(y, y^*) \langle \mathbf{w}, \phi(x, y^*) - \phi(x, y) \rangle \geq 0$. The second inequality follows by

definition of the loss.

We conclude this section with a simple lemma which is at the heart of several derivations of Joachims [2005].

Lemma A.1.1. *Denote by $\delta(y, y')$ a label loss and let $\phi(x_i, y_i)$ be a feature map for observations (x_i, y_i) with $1 \leq i \leq m$. Moreover, denote by X, Y the set of all m inputs and labels respectively. Finally let*

$$\Phi(X, Y) := \sum_{i=1}^{m} \phi(x_i, y_i) \text{ and } \Delta(Y, Y') := \sum_{i=1}^{m} \delta(y_i, y_i'). \tag{A.9}$$

Then the following two losses are equivalent:

$$\sum_{i=1}^{m} \max_{y'} \langle \mathbf{w}, \phi(x_i, y') - \phi(x_i, y_i) \rangle + \delta(y_i, y') \text{ and}$$
$$\max_{Y'} \langle \mathbf{w}, \Phi(X, Y') - \Phi(X, Y) \rangle + \Delta(Y, Y').$$

This is immediately obvious, since both feature map and label loss decompose, which allows us to perform maximization over Y' by maximizing each of its m components. In doing so, we showed that aggregating all data and labels into a single feature map and loss yields results identical to minimizing the sum over all individual losses. This holds, in particular, for the sample error loss of Joachims [2005].

A.1.1 Intractable Models

We now discuss cases where computing $l(x, y, \mathbf{w})$ itself is too expensive. For instance, in intractable graphical models, the computation of $\sum_y \exp \langle \mathbf{w}, \phi(x, y) \rangle$ cannot be computed efficiently. Wainwright and Jordan [2003] proposed the use of a convex majorization of the log-partition function in those cases. In our setting this means that instead of dealing with

$$l(x, y, \mathbf{w}) = g(\mathbf{w}\,|x) - \langle \mathbf{w}, \phi(x, y) \rangle \text{ where } g(\mathbf{w}\,|x) := \log \sum_y \exp \langle \mathbf{w}, \phi(x, y) \rangle \tag{A.10}$$

one uses a more easily computable convex upper bound on g via

$$\sup_{\boldsymbol{\mu} \in \mathrm{MARG}(x)} \langle \mathbf{w}, \boldsymbol{\mu} \rangle + H_{\mathrm{Gauss}}(\boldsymbol{\mu}|x). \tag{A.11}$$

Here $\mathrm{MARG}(x)$ is an outer bound on the conditional marginal polytope associated with the map $\phi(x, y)$. Moreover, $H_{\mathrm{Gauss}}(\boldsymbol{\mu}|x)$ is an upper bound on the entropy by using a Gaussian with identical variance. There also exist more refined tree decompositions. The key benefit of our approach is that the solution $\boldsymbol{\mu}$ of the optimization problem

(A.11) can immediately be used as a gradient of the upper bound. This is rather computationally efficient.

Likewise note that Taskar et al. [2004] used relaxations when solving structured prediction problems of the form

$$l(x, y, \mathbf{w}) = \max_{y'} \Gamma(y, y') \left\langle \mathbf{w}, \phi(x, y') - \phi(x, y) \right\rangle + \Delta(y, y'),$$

by enlarging the domain of maximization with respect to y'. For instance, instead of an integer programming problem we might relax the setting to a linear program which is much cheaper to solve. This, again, provides an upper bound on the original loss function.

A.1.2 Ontologies

Assume that the labels we want to estimate can be found to belong to a directed acyclic graph (DAG). For instance, this may be a gene-ontology graph [Ashburner et al., 2000] a patent hierarchy [Cai and Hofmann, 2004], or a genealogy. In these cases we have a hierarchy of categories to which an input $x \in \mathbb{R}^d$ may belong to. Figure A.1 gives two examples of such directed acyclic graphs. The first example is a binary tree, while the second contains nodes with different numbers of children (*e.g.*, node 4 and 12), nodes at different levels having children (*e.g.*, nodes 5 and 12), and nodes which have more than one parent (*e.g.*, node 5).

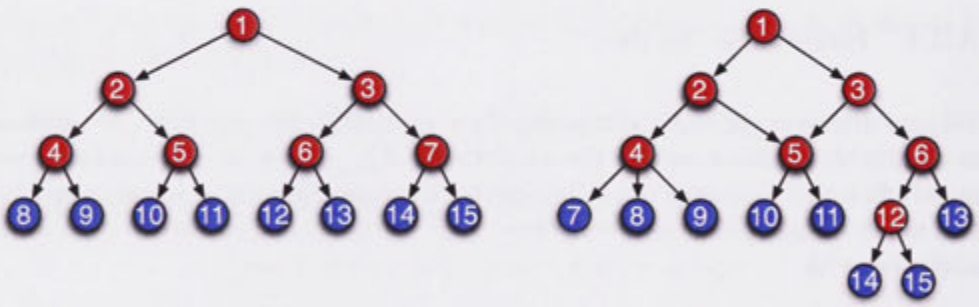


Figure A.1: Two ontologies. **Left:** a binary hierarchy with internal nodes $\{1, \ldots, 7\}$ and labels $\{8, \ldots 15\}$. **Right:** a generic directed acyclic graph with internal nodes $\{1, \ldots, 6, 12\}$ and labels $\{7, \ldots, 11, 13, \ldots, 15\}$. Note that node 5 has two parents, namely nodes 2 and 3. Moreover, the labels need not be found at the same level of the tree: nodes 14 and 15 are one level lower than the rest of the nodes.

The goal here is to build a classifier which is able to categorize inputs according to which leaf node they belong to (each leaf node is assigned a label y). Denote by $k + 1$ the number of nodes in the DAG including the root node. In this case we may design

a feature map $\phi(y) \in \mathbb{R}^k$ [Cai and Hofmann, 2004] by associating with every label y the vector describing the path from the root node to y, ignoring the root node itself. For instance, for the first DAG in Figure A.1 we have

$$\phi(8) = (1,0,1,0,0,0,1,0,0,0,0,0,0,0) \text{ and}$$
$$\phi(13) = (0,1,0,0,1,0,0,0,0,0,0,1,0,0).$$

Whenever several paths are admissible, as in the right DAG of Figure A.1 we average over all possible paths. For example, we have

$$\phi(10) = (0.5,0.5,0,1,0,0,0,0,1,0,0,0,0,0) \text{ and}$$
$$\phi(15) = (0,1,0,0,1,0,0,0,0,0,0,1,0,0,1).$$

Also note that the lengths of the paths need not be the same (*e.g.*, to reach 15 it takes a longer path than to reach 13). Likewise, it is natural to assume that $\Delta(y, y')$, *i.e.*, the cost for mislabeling y as y' will depend on the similarity of the path. In other words, it is likely that the cost for placing x into the wrong sub-sub-category is less than getting the main category of the object wrong.

To complete the setting, note that for

$$\phi(x, y) = \phi(y) \otimes x \in \mathbb{R}^{dk}$$

the cost of computing all labels is k inner products, since the value of $\langle \mathbf{w}, \phi(x, y) \rangle$ for a particular y can be obtained by the sum of the contributions for the segments of the path. This means that the values for *all* terms can be computed by a simple breadth first traversal through the graph.

Also note that $\phi(y) - \phi(y')$ is nonzero only for those edges where the paths for y and y' differ. Hence we only change weights on those parts of the graph where the categorization differs. Algorithm 11 describes the loss and subgradient computation for the max-margin type of loss function.

The same reasoning applies to the logistic type of loss function. The only difference is that we need to compute a *soft-max* over paths rather than exclusively choosing the best path over the ontology. Again, a breadth-first recursion suffices: each of the leaves y of the DAG is associated with a probability $p(y|x)$. To obtain $\mathbf{E}_{y \sim p(y|x)} [\phi(y)]$ all we need to do is perform a bottom-up traversal of the DAG summing over all probability weights on the path. Wherever a node has more than one parent, we distribute the probability weight equally over its parents.

Algorithm 11 Ontology Loss

input: Examples $\{(x_i, y_i)\}_{i=1}^m \subset \mathbb{R}^d \times \mathbb{R}^k$, weight vector $\mathbf{w} \in \mathbb{R}^{dk}$
initialization: $\mathbf{a} = \mathbf{0} \in \mathbb{R}^{dk}$ and $r = 0$
for $i = 1$ **to** m **do**
  Let $f_j = \langle \phi(x_i, e_j), \mathbf{w} \rangle$, $\forall j \in [k]$ where e_j is a d-dimensional vector with all components being zero but the j-th component being 1
  Let D_i be the DAG with edges annotated with the values of f_j
  Traverse D_i to find a path y^* that maximizes the sum of values on the path plus the value of $\Delta(y_i, y^*)$
  $\mathbf{a} \leftarrow \mathbf{a} + \phi(x_i, y^*) - \phi(x_i, y_i)$
  $r \leftarrow r + \langle \mathbf{w}, \phi(x_i, y^*) - \phi(x_i, y_i) \rangle + \Delta(y_i, y^*)$
end for
return: Risk r and subgradient $\mathbf{a}$

A.2 Loss/Risk for Multivariate Performance Scores

We now discuss some examples of structured loss functions reminiscent of the works by Joachims [2005, 2006] that upper bound various multivariate performance scores.

A.2.1 Preference Relations

In general, this loss may be described by means of a set of preference relations $j \succeq i$ for arbitrary pairs $(i, j) \in [m]^2$ associated with a cost $C(i, j)$ which is incurred whenever i is ranked above j. This set of preferences may or may not form a partial or a total order on the domain of all observations. In these cases efficient computations along the lines of Joachims [2005, 2006] exist. In general, this is not the case and we need to rely on the fact that the set P containing all preferences is sufficiently small that it can be enumerated efficiently. The risk is then given by

$$\frac{1}{|P|} \sum_{(i,j) \in P} C(i, j) \mathbf{I}(\langle \mathbf{w}, x_i \rangle > \langle \mathbf{w}, x_j \rangle) \tag{A.12}$$

Again, the same majorization argument as before allows us to write a convex upper bound

$$R(\mathbf{w}) = \frac{1}{|P|} \sum_{(i,j) \in P} C(i, j) \max(0, 1 + \langle \mathbf{w}, x_i \rangle - \langle \mathbf{w}, x_j \rangle) \tag{A.13}$$

$$\text{where } \partial_{\mathbf{w}} R(\mathbf{w}) = \frac{1}{|P|} \sum_{(i,j) \in P} C(i, j) \begin{cases} 0 & \text{if } \langle \mathbf{w}, x_j - x_i \rangle \geq 1 \\ x_i - x_j & \text{otherwise} \end{cases} \tag{A.14}$$

The implementation is straightforward, as given in Algorithm 12.

Algorithm 12 Preference Relations Risk

input: Examples $\{(x_i, y_i)\}_{i=1}^m$, weight vector $\mathbf{w}$, cost matrix C, preference set P
initialization: $\mathbf{f} = \mathbf{0}_m$, $r = 0$, and $\mathbf{a} = \mathbf{0}_d$
$f_i \leftarrow \langle \mathbf{w}, x_i \rangle$, $\forall i \in [m]$
while $(i, j) \in P$ **do**
  if $f_j - f_i < 1$ **then**
    $r \leftarrow r + C(i, j)(1 + f_i - f_j)$
    $\mathbf{a} \leftarrow \mathbf{a} + C(i, j)(x_i - x_j)$
  end if
end while
return Risk r and subgradient $\mathbf{a}$

A.2.2 Ranking

In document (and webpage) ranking we are often in a situation similar to the ordinal regression [Joachims, 2006], however with the difference that we do not only care about objects x_i being ranked according to label y_i but moreover that different degrees of importance are placed on different documents.

The information retrieval literature is full with a large number of different scoring functions. Examples are criteria such as *Normalized Discounted Cumulative Gain (NDCG)*, *Mean Reciprocal Rank (MRR)*, *Precision@n*, or *Expected Rank Utility (ERU)*. They are used to address the issue of evaluating rankers, search engines or recommender sytems [Voorhees, 2001, Jarvelin and Kekalainen, 2002, Breese et al., 1998, Basilico and Hofmann, 2004]. For instance, in webpage ranking only the first k retrieved documents that matter, since users are unlikely to look beyond the first k, say 10, retrieved webpages in an internet search. Chapelle et al. [2007] show that these scores can be optimized directly by minimizing the following loss:

$$l(x, y, \mathbf{w}) = \max_{\pi} \sum_{i}^{k} c_i \langle \mathbf{w}, \bar{x}_{\pi_i} - \bar{x}_i \rangle + \langle \mathbf{p} - \mathbf{p}(\pi), \mathbf{q}(y) \rangle , \tag{A.15}$$

where $x := (\bar{x}_1, \ldots, \bar{x}_k) \in \mathbb{R}^{d \times k}$ is a set of k documents $\bar{x}$, $y \in \mathbb{R}^k$ is its relevance vector, the documents are assumed to be arranged in order of decreasing relevance, $\mathbf{c} := (c_1, \ldots, c_k)$ with $c_1 > \cdots > c_k$, π is a permutation of $[k]$, the vectors $\mathbf{p}$ and $\mathbf{q}(y)$ depend on the choice of a particular ranking measure, and $\mathbf{p}(\pi)$ denotes the permutation of $\mathbf{p}$ according to π. Let $\mathbf{f} := (f_1, \ldots, f_m)$ where $f_i := \langle \mathbf{w}, \bar{x}_i \rangle$, we may rewrite (A.15) as

$$l(x, y, \mathbf{w}) = \max_{\pi} \left[\mathbf{c}^\top \mathbf{f}(\pi) - \mathbf{p}(\pi)^\top \mathbf{q}(y) \right] - \mathbf{c}^\top \mathbf{f} + \mathbf{p}^\top \mathbf{q}(y)$$

Algorithm 13 Ranking Loss

1: **input:** Example (x, y), vector $\mathbf{c}$, and weight vector $\mathbf{w}$
2: Compute vectors $\mathbf{p}$ and $\mathbf{q}(y)$ according to some ranking measure
3: Let $f_i = \langle \mathbf{w}, \bar{x}_i \rangle$, $\forall i \in [k]$
4: Compute elements of matrix $C_{ij} = c_i f_j - q_i p_j$
5: Obtain π by solving a linear assignment problem with cost matrix C
6: Compute loss $l \leftarrow \mathbf{c}^\top(\mathbf{f}(\pi) - \mathbf{f}) + (\mathbf{p} - \mathbf{p}(\pi))^\top \mathbf{q}(y)$
7: Compute subgradient $\mathbf{a} = \sum_{i=1}^{k} (c_{\pi_i^{-1}} - c_i)\bar{x}_i$
8: **return:** Loss l and subgradient $\mathbf{a}$

and consequently the derivative of $l(x, y, \mathbf{w})$ with respect to $\mathbf{w}$ is given by

$$\partial_{\mathbf{w}} l(x, y, \mathbf{w}) = \sum_{i=1}^{m} (c_{\bar{\pi}_i^{-1}} - c_i)\bar{x}_i \text{ where } \bar{\pi} = \operatorname*{argmax}_{\pi} \mathbf{c}^\top \mathbf{f}(\pi) - \mathbf{p}(\pi)^\top \mathbf{q}(y). \tag{A.16}$$

Here π^{-1} denotes the inverse permutation, such that $\pi \circ \pi^{-1}$ is an identity. Finding the permutation $\bar{\pi}$ in (A.16) is a linear assignment problem (LAP) which can be easily solved by the Hungarian Marriage algorithm, that is, the Kuhn-Munkres algorithm.

The original papers by Kuhn [1955] and Munkres [1957] implied an algorithm with $O(k^3)$ cost in the number of terms. Later, Karp [1980] suggested an algorithm with expected quadratic time in the size of the assignment problem (ignoring log-factors). Finally, Orlin and Lee [1993] proposed a linear time algorithm for large problems. Algorithm 13 spells out the steps in computing the ranking loss and its subgradient.

Proofs

B.1 Inequalities for Recursive Sequences

The following are some inequalities for recursively defined sequences.

Lemma B.1.1 ([Abe et al., 2001, Sublemma 5.4]). *Let $\langle p_1, p_2, \ldots \rangle$ be a sequence of non-negative numbers satisfying the following recurrence, for $t \geq 1$: $p_t - p_{t+1} \geq z(p_t)^2$, where $z > 0$ is a constant. Then for all integers $t \geq 1$,*

$$p_t \leq \frac{1}{z(t - 1 + \frac{1}{p_1 z})}.$$

Furthermore $p_t \leq \epsilon$ whenever

$$t \geq \frac{1}{z}\left(\frac{1}{\epsilon} - \frac{1}{p_1}\right) + 1.$$

Lemma B.1.2. *Let $\langle p_1, p_2, \ldots \rangle$ be a sequence of non-negative numbers satisfying the following recurrence, for $t \geq 1$: $p_{t+1} \leq zp_t$, where $z \in (0, 1)$ is a constant. Then for all integers $t \geq 1$,*

$$p_t \leq z^{t-1} p_1.$$

Furthermore $p_t \leq \epsilon$ whenever

$$t \geq \log(p_1/\epsilon)/\log(1/z) + 1.$$

Proof. Note that $p_{t+1} \leq zp_t \leq z^2 p_{t-1} \leq \ldots \leq z^t p_1$ implies that $p_t \leq z^{t-1} p_1$. This proves the first part. The second part is proved by setting $p_t = \epsilon$, solving $\epsilon \leq z^{t-1} p_1$ for t, and rearranging the terms. □

B.2 Proof sketch for Theorem 2.3.1

We sketch a proof for Theorem 2.3.1 by combining the results of Robinson [1999], Kiwiel [2000], and Belloni [2005]. For the analysis of null steps iteration bound, we follow mostly the approach of Belloni [2005] which is a simplified version of that in Kiwiel [2000] (*i.e.*, keeping only the dominant term). We first recall some quantities that will be handy in the proof:

$$\begin{aligned} B_t &\subseteq \{1, \ldots, t\}, \\ \check{J}_t(\mathbf{w}) &:= \max_{i \in B_t} \{J(\mathbf{w}_i) + \langle \mathbf{w} - \mathbf{w}_i, \mathbf{g}_i \rangle\}, \\ \mathring{J}_t(\mathbf{w}) &:= \check{J}_t(\mathbf{w}) + \tfrac{\mu_t}{2} \|\mathbf{w} - \hat{\mathbf{w}}_t\|_2^2, \\ \mathbf{w}_{t+1} &:= \operatorname{argmin}_{\mathbf{w}} \mathring{J}_t(\mathbf{w}), \\ \epsilon_t &:= J(\hat{\mathbf{w}}_t) - \check{J}_t(\mathbf{w}_{t+1}), \\ \delta_t &:= J(\hat{\mathbf{w}}_t) - \mathring{J}_t(\mathbf{w}_{t+1}), \end{aligned}$$

and c_t the Lipschitz constant of J and $\check{J}_t$ in the neighborhood $U_t := \{\mathbf{w} : \|\mathbf{w} - \hat{\mathbf{w}}_t\| \leq \delta_t / \mu_{\min}\}$.

Assume that Algorithm 1 just finished a serious step and obtained the prox-center $\hat{\mathbf{w}}_t$. Now we proceed to show the number n of null steps required before the serious step condition (*cf.* (2.24))

$$J(\hat{\mathbf{w}}_{t+n}) - J(\mathbf{w}_{t+n+1}) \geq \rho \epsilon_{t+n},$$

is satisfied. From part (ii) of Lemma 3.1 in Kiwiel [2000], we have that

$$\epsilon_t \geq \delta_t \geq \epsilon_t / 2.$$

The leftmost inequality is obvious by definition while the rightmost is obtained after some algebra using the fact that $\mu_t(\mathbf{w}_{t+1} - \hat{\mathbf{w}}_t) \in \partial \check{J}_t(\mathbf{w}_{t+1})$. Furthermore, from the proof of Theorem 10.2 in Belloni [2005] we see that

$$\delta_i - \delta_{i+1} \geq \frac{\mu_{\min}(1-\rho)^2}{8c_t^2} \delta_i^2, \ i \geq t, \tag{B.1}$$

It follows that the null steps generating the strictly decreasing sequence $(\delta_{t+i})_{i \geq 1}$ must stop before the algorithm terminates *i.e.*, $\delta_{t+n} \leq 2\epsilon$.

Applying Lemma B.1.1 to (B.1) and setting $\delta_{t+n} = 2\epsilon$, we have that the number n of null steps required before a serious step is met or the algorithm terminates is

$$n \geq \frac{8c_t^2}{\mu_{\min}(1-\rho)^2} \left(\frac{1}{2\epsilon} - \frac{1}{\delta_t} \right) + 1 \tag{B.2}$$

This concludes the null steps analysis.

We now proceed to the analysis of serious steps. For notational convenience, we denote by $\mathbf{w}_{(1)}, \mathbf{w}_{(2)}, \mathbf{w}_{(3)}, \ldots$ the iterates that triggered serious steps: $\mathbf{w}_{k_1}, \mathbf{w}_{k_2}, \mathbf{w}_{k_3}, \ldots$ where $k_1 < k_2 < k_3 < \ldots$. By default, we let the first iteration be a serious step *i.e.*, $\mathbf{w}_1 = \mathbf{w}_{(1)}$. By the proof of Theorem 3 in Robinson [1999] and an application of Theorem C.0.11 (which links the modulus of strong convexity of J to the Lipschitz constant of the gradient mapping of conjugate J^* of J), we obtain

$$\epsilon_{(s+1)} \leq \frac{1-\rho}{1+\mu_{\max}\sigma^{-1}}\epsilon_{(s)}. \tag{B.3}$$

Since $0 < (1-\rho)/(1+\mu_{\max}\sigma^{-1}) < 1$, we can apply Lemma B.1.2 to (B.3) and obtain the number s of serious steps which leads to the last sequence of null steps or termination of algorithm,

$$\begin{aligned} s &\geq \log\left(\frac{\epsilon_1}{\epsilon}\right) \Big/ \log\left(\frac{1+\mu_{\max}\sigma^{-1}}{1-\rho}\right) + 1 \\ &= \log\left(\frac{\epsilon_1(1+\mu_{\max}\sigma^{-1})}{\epsilon(1-\rho)}\right) \Big/ \log\left(\frac{1+\mu_{\max}\sigma^{-1}}{1-\rho}\right) \end{aligned} \tag{B.4}$$

Multiplying (B.2) and (B.4) we see that Algorithm 1 terminates with an ϵ-accurate solution $\mathbf{w}_T$ after

$$\begin{aligned} T &\leq \left[1 + \frac{8c_t^2}{\mu_{\min}(1-\rho)^2}\left(\frac{1}{2\epsilon} - \frac{1}{\delta_t}\right)\right] \log\left(\frac{\epsilon_1(1+\mu_{\max}\sigma^{-1})}{\epsilon(1-\rho)}\right) \Big/ \log\left(\frac{1+\mu_{\max}\sigma^{-1}}{1-\rho}\right) \\ &\leq \left[1 + \frac{4c^2}{\epsilon\mu_{\min}(1-\rho)^2}\right] \log\left(\frac{\epsilon_1(1+\mu_{\max}\sigma^{-1})}{\epsilon(1-\rho)}\right) \end{aligned}$$

iterations, where $c := \max_{i\in[T]} c_i$, as claimed.

B.3 Proof of Theorem 3.2.1

To show Theorem 3.2.1 we need several technical intermediate steps. Let $\gamma_t := J(\mathbf{w}_{t+1}) - J_t(\mathbf{w}_{t+1})$ and recall that $\epsilon_t := \min_{i\in[t+1]} J(\mathbf{w}_i) - J_t(\mathbf{w}_{t+1})$. Also let $\mathbf{w}^* := \operatorname{argmin}_{\mathbf{w}} J(\mathbf{w})$. The following lemma establishes some useful properties of γ_t and ϵ_t.

Lemma B.3.1. *We have*

$$J_i(\mathbf{w}_{i+1}) \leq J_t(\mathbf{w}_{t+1}) \leq J(\mathbf{w}^*) \leq J(\mathbf{w}_{t+1}) = J_{t+1}(\mathbf{w}_{t+1}) \quad \textit{for all} \quad i \leq t.$$

Furthermore, ϵ_t is monotonically decreasing with

$$\epsilon_t - \epsilon_{t+1} \geq J_{t+1}(\mathbf{w}_{t+2}) - J_t(\mathbf{w}_{t+1}) \geq 0. \tag{B.5}$$

Also, ϵ_t upper bounds the distance from optimality via

$$\gamma_t \geq \epsilon_t \geq \min_{i \in [t+1]} J(\mathbf{w}_i) - J(\mathbf{w}^*). \tag{B.6}$$

Proof. Since $J_i(\mathbf{w}) \leq J_t(\mathbf{w}) \leq J(\mathbf{w})$ for all $\mathbf{w} \in \mathbb{R}^d$ and for all $i \leq t$, this property also applies to their respective minima. Moreover, since $\mathbf{w}^*$ minimizes $J(\mathbf{w})$ we have $J(\mathbf{w}^*) \leq J(\mathbf{w}_t)$. Since Taylor expansions are exact at the point of expansion $J(\mathbf{w}_t) = J_t(\mathbf{w}_t)$. The inequalities (B.5) follow immediately from the definition of ϵ_t and the fact that $\min_{\mathbf{w}} J_t(\mathbf{w})$ is non-decreasing for all $t > 0$. Finally, the inequalities (B.6) hold by the definitions of γ_t and ϵ_t as well as the fact that $J(\mathbf{w}^*) \geq \min_{\mathbf{w}} J_t(\mathbf{w})$ for all $t > 0$. □

Our second technical lemma allows us to bound the maximum value of a concave function provided that we know its first derivative and a bound on the second derivative.

Lemma B.3.2. *Denote by $h : [0, 1] \to \mathbb{R}$ a concave function with $h(0) = 0$, $\nabla h(0) = c_1$, and $|\nabla^2(w)| \leq c_2 \; \forall w \in [0, 1]$. Then we have $\max_{w \in [0,1]} h(w) \geq \frac{c_1}{2} \min(\frac{c_1}{c_2}, 1)$.*

Proof. We first observe that $g(w) := c_1 w - \frac{c_2}{2} w^2 \leq h(w) \; \forall w$ implies $\max_{w \in [0,1]} h(w) \geq \max_{w \in [0,1]} g(w)$. g attains the unconstrained maximum $\frac{c_1^2}{2c_2}$ at $w = \frac{c_1}{c_2}$. Since g is monotonically increasing in $[0, \frac{c_1}{c_2}]$, if $c_1 > c_2$ we pick $w = 1$ which yields constrained maximum $c_1 - \frac{c_2}{2} > \frac{c_1}{2}$. Taking the minimum over both maxima proves the claim.

□

To apply the above result, we need to compute the gradient and Hessian of $J^*_{t+1}(\boldsymbol{\alpha})$ with respect to the search direction $((1 - \eta)\boldsymbol{\alpha}_t, \eta)$. The following lemma takes care of the gradient:

Lemma B.3.3. *Denote by $\boldsymbol{\alpha}_t$ the solution of (3.4) at t-th iteration. Moreover, denote by $\mathbf{A}_{t+1} := [\mathbf{A}_t, \mathbf{a}_{t+1}]$ and $\mathbf{b}_{t+1} := [\mathbf{b}_t, b_{t+1}]$ the extended matrices and vectors needed to define the dual problem for $(t+1)$-th iteration, and let $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^{t+1}$. Then the following holds:*

$$\partial J^*_{t+1}([\boldsymbol{\alpha}_t, 0]) = \mathbf{A}_{t+1}^\top \mathbf{w}_{t+1} + \mathbf{b}_{t+1} \quad \textit{and} \tag{B.7}$$

$$[-\boldsymbol{\alpha}_t, 1]^\top \left[\mathbf{A}_{t+1}^\top \mathbf{w}_{t+1} + \mathbf{b}_{t+1} \right] = J_{t+1}(\mathbf{w}_{t+1}) - J_t(\mathbf{w}_{t+1}) = \gamma_t \tag{B.8}$$

Proof. By the dual connection $\partial \Omega^*(-\lambda^{-1} \mathbf{A}_t \boldsymbol{\alpha}_t) = \mathbf{w}_{t+1}$. Hence we have that $\partial_{\bar{\boldsymbol{\alpha}}}[-\lambda \Omega^*(-\lambda^{-1} \mathbf{A}_{t+1} \bar{\boldsymbol{\alpha}}) + \bar{\boldsymbol{\alpha}}^\top \mathbf{b}_{t+1}] = \mathbf{A}_{t+1}^\top \mathbf{w}_{t+1} + \mathbf{b}_{t+1}$ for $\bar{\boldsymbol{\alpha}} = [\boldsymbol{\alpha}_t, 0]^\top$. This proves the first claim. To see the second part we eliminate ξ from of the Lagrangian (3.5) and

write the partial Lagrangian

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \lambda\Omega(\mathbf{w}) + \boldsymbol{\alpha}^\top \left(\mathbf{A}_t^\top \mathbf{w} + \mathbf{b}_t\right) \text{ with } \boldsymbol{\alpha} \geq \mathbf{0}_t.$$

The result follows by noting that at optimality $L(\mathbf{w}_{t+1}, \boldsymbol{\alpha}_t) = J_t(\mathbf{w}_{t+1})$ and $J_{t+1}(\mathbf{w}_{t+1}) = \lambda\Omega(\mathbf{w}_{t+1}) + \langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1} \rangle + b_{t+1}$. Consequently we have

$$J_{t+1}(\mathbf{w}_{t+1}) - J_t(\mathbf{w}_{t+1}) = \lambda\Omega(\mathbf{w}_{t+1}) + \langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1} \rangle + b_{t+1} - \lambda\Omega(\mathbf{w}_{t+1}) - \boldsymbol{\alpha}_t^\top (\mathbf{A}_t^\top \mathbf{w}_{t+1} + \mathbf{b}_t)$$

Rearranging terms proves the claim. □

To apply Lemma B.3.2 we also need to bound the (generalized) Hessian of $J_{t+1}^*(\boldsymbol{\alpha})$. By assumption, Ω is σ-strongly convex, hence Ω^* is continuously differentiable and has σ^{-1}-Lipschitz continuous gradient by Theorem C.0.11. However, it is not necessary that Ω^* is twice differentiable. If Ω^* is twice differentiable, we let $\partial^2\Omega^*(\boldsymbol{\mu})$ to denote the Hessian (*i.e.*, $\nabla^2\Omega^*(\boldsymbol{\mu})$) of Ω^* at $\boldsymbol{\mu}$. Otherwise, $\partial^2\Omega^*(\boldsymbol{\mu})$ denotes the generalized Hessian [Hiriart-Urruty et al., 1984] of Ω^* *i.e.*, a set of matrices defined as the convex hull of the set

$$\left\{\mathbf{M} \ : \ \exists\, \mathbf{v} \to \boldsymbol{\mu} \text{ with } \Omega^* \text{ twice differentiable at } \mathbf{v} \text{ and } \nabla^2\Omega^*(\mathbf{v}) \to \mathbf{M}\right\}. \tag{B.9}$$

Arguably, the generalization of Hessian for continuously differentiable functions with Lipschitz continuous gradient is analogous to the generalization of gradient for (continuous but) non-differentiable functions. We note the following lemma which will be handy in the proof of Theorem 3.2.1.

Lemma B.3.4. *For $\Omega : \mathbb{R}^d \to \mathbb{R}$ a σ-strongly convex function, we have, for any $\mathbf{u} \in \mathbb{R}^d$,*

$$\|\mathbf{M}\| \leq \sigma^{-1}$$

where $\mathbf{M} \in \partial^2\Omega^*(\mathbf{u})$, and Ω^* is the conjugate of Ω.

Proof. By Theorem C.0.11, we know that Ω^* has σ^{-1}-Lipschitz continuous gradient. Then by the definition of Lipschitz continuous gradient (see Definition C.0.8), the inequality holds for all points where the Hessian $\nabla^2\Omega^*$ exists. At points where Hessian is not defined but the generalized Hessian $\partial^2\Omega^*$, we know from (B.9) that any element of $\partial^2\Omega^*$ is a convex combination of some Hessian matrices at points where $\nabla^2\Omega^*$ exits, hence by the convexity of norm, the inequality holds. Putting the two cases together we conclude the proof. □

Now we are in a position to prove the bounds on the (generalized) Hessian of J_{t+1}^* and then Theorem 3.2.1.

Lemma B.3.5. *Under the assumptions of Lemma B.3.3 we have*

$$\partial^2 J^*_{t+1}(\bar{\boldsymbol{\alpha}}) = -\lambda^{-1}\,\mathbf{A}^\top_{t+1}\,\partial^2\Omega^*(-\lambda^{-1}\,\mathbf{A}_{t+1}\,\bar{\boldsymbol{\alpha}})\,\mathbf{A}_{t+1},\ \textit{and} \tag{B.10}$$

$$\mathbf{A}_{t+1}[-\boldsymbol{\alpha}_t, 1] = \mathbf{g}_{t+1} \in \partial J(\mathbf{w}_{t+1}) \tag{B.11}$$

Proof. The first equality is immediate from the chain rule. Next note that $\partial\Omega(\mathbf{w}_{t+1}) = -\lambda^{-1}\,\mathbf{A}_t\,\boldsymbol{\alpha}_t$ by dual connection. Since $\mathbf{a}_{t+1} \in \partial R(\mathbf{w}_{t+1})$ the claim follows from $J(\mathbf{w}) = \lambda\Omega(\mathbf{w}) + R(\mathbf{w})$. □

This result allows us to express the second derivative of the dual objective function (3.4) in terms of the gradient of the risk functional. The idea is that as we approach optimality, the second derivative will vanish. We will use this fact to argue that for continuously differentiable risks, Algorithm 2 exhibits linear rate of convergence.

Proof. [Theorem 3.2.1] We overload the notation for J^*_{t+1} by defining the following one dimensional concave function

$$\begin{aligned} J^*_{t+1}(\eta) &:= J^*_{t+1}([(1-\eta)\boldsymbol{\alpha}_t, \eta]) \\ &= -\lambda\Omega^*(-\lambda^{-1}\,\mathbf{A}_{t+1}[(1-\eta)\boldsymbol{\alpha}_t^\top, \eta]) + [(1-\eta)\boldsymbol{\alpha}_t^\top, \eta]\,\mathbf{b}_{t+1}\,. \end{aligned}$$

Clearly, $J^*_{t+1}(0) = J_t(\mathbf{w}_{t+1})$. Furthermore, by (B.8), (B.10), and (B.11) it follows that

$$\begin{aligned} \partial J^*_{t+1}(\eta)|_{\eta=0} &= [-\boldsymbol{\alpha}_t, 1]^\top J^*_{t+1}([\boldsymbol{\alpha}_t, 0]) = \gamma_t, \text{ and} \\ \partial^2 J^*_{t+1}(\eta) &= -\lambda^{-1}[-\boldsymbol{\alpha}_t, 1]^\top\,\mathbf{A}^\top_{t+1}\,\mathbf{M}\,\mathbf{A}_{t+1}[-\boldsymbol{\alpha}_t, 1] \\ &= -\lambda^{-1}\,\mathbf{g}^\top_{t+1}\,\mathbf{M}\,\mathbf{g}_{t+1} \\ &=: r \end{aligned}$$

where $\mathbf{M} \in \partial^2\Omega^*(-\lambda^{-1}\,\mathbf{A}_{t+1}[(1-\eta)\boldsymbol{\alpha}_t, \eta])$. By Lemma B.3.4, we have that $\|\mathbf{M}\| \leq \sigma^{-1}$, hence,

$$|r| \leq \left\|\mathbf{g}_{t+1}\right\|^2/(\lambda\sigma).$$

Next we need to bound the gradient of J. For this purpose note that $\partial\lambda\Omega(\mathbf{w}_{t+1}) = -\,\mathbf{A}_t^\top\,\boldsymbol{\alpha}_t$ and that $\|\boldsymbol{\alpha}_t\|_1 = 1$. This implies that $\partial\lambda\Omega(\mathbf{w}_{t+1})$ lies in the convex hull of the past gradients, $\mathbf{a}_i$. By our assumption on $\|\mathbf{a}_i\|$ it follows that $\|\partial\lambda\Omega(\mathbf{w}_{t+1})\| \leq G$. We conclude that

$$\left\|\mathbf{g}_{t+1}\right\|^2 \leq 4G^2 \quad \text{and} \quad |r| \leq 4G^2/(\lambda\sigma).$$

Invoking Lemma B.3.2 on $J^*_{t+1}(\eta) - J_t(\mathbf{w}_{t+1})$ shows that

$$J^*_{t+1}(\eta) - J_t(\mathbf{w}_{t+1}) \geq \frac{\gamma_t}{2}\min(1, \lambda\sigma\gamma_t/(4G^2)).$$

We now upper bound the LHS of the above inequality as follows:

$$\epsilon_t - \epsilon_{t+1} \geq J_{t+1}(\mathbf{w}_{t+2}) - J_t(\mathbf{w}_{t+1}) \geq J^*_{t+1}(\eta) - J_t(\mathbf{w}_{t+1}) \geq \frac{\gamma_t}{2} \min(1, \lambda\sigma\gamma_t/(4G^2)). \tag{B.12}$$

The first inequality follows from Lemma B.3.1 while the second follows by observing that $J_{t+1}(\mathbf{w}_{t+2}) = J^*_{t+1}(\boldsymbol{\alpha}_{t+1}) \geq J^*_{t+1}(\eta)$. The RHS of the third inequality on the other hand can be lower bounded by observing that $\gamma_t \geq \epsilon_t$, which follows from Lemma B.3.1. This in turn obtains (3.6).

Now we prove the second part of the theorem where J is assumed to be continuously differentiable and has H-Lipschitz continuous gradient. Note that (3.6) already yields the $\epsilon_t/2$ decrease when $\epsilon_t \geq 4G^2/(\lambda\sigma)$. To show the other parts we need to show that the gradient of J vanishes as the algorithm converges to the optimal solution. Towards this end, we apply Lemma B.3.2 in the *primal*.[1] This allows us to bound $\|\nabla J(\mathbf{w}_{t+1})\|$ in terms of γ_t: Plugging in the first and second derivative of $J(\mathbf{w}_{t+1})$ and noting that, by Definition C.0.8, the norm of the (generalized) Hessian of J is bounded from above by H, we obtain

$$\gamma_t \geq \frac{1}{2} \|\nabla J(\mathbf{w}_{t+1})\| \min(1, \|\nabla J(\mathbf{w}_{t+1})\| / H).$$

If $\|\nabla J(\mathbf{w}_{t+1})\| > H$, then $\gamma_t \geq \frac{1}{2}\|\nabla J(\mathbf{w}_{t+1})\|$ which in turn yields $|r| \leq 4\gamma_t^2/(\lambda\sigma)$. Plugging this into Lemma B.3.2 yields a lower bound on the improvement of $\lambda\sigma/8$.

Finally, for $\|\nabla J(\mathbf{w}_{t+1})\| \leq H$ we have $\gamma_t \geq \|\nabla J(\mathbf{w}_{t+1})\|^2 / (2H)$, which implies $|r| \leq 2H\gamma_t/(\lambda\sigma)$. Plugging this into Lemma B.3.2 yields an improvement of $\lambda\sigma\gamma_t/(4H) \geq \lambda\sigma\epsilon_t/(4H)$.

Since both cases cover the remaining range of convergence, the minimum $\min(\lambda\sigma/8, \lambda\sigma\epsilon_t/(4H))$ provides a lower bound for the improvement. The crossover point between both terms occurs at $\epsilon_t = H/2$. Rearranging the conditions leads to the (pessimistic) improvement guarantees of the second claim. □

Note that a key step in the above analysis involved bounding $r := \partial_\eta^2 J^*_{t+1}(\eta)$. For a number of regularizers tighter bounds can be obtained. The following bounds are essentially due to Shalev-Shwartz and Singer [2006]:

- For squared L_p norm regularization, *i.e.*, $\Omega^*(\boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{\mu}\|_q^2$ we have $r \leq (q-1)\left\|\mathbf{g}_{t+1}\right\|_q^2$ where $\mathbf{g}_{t+1} \in \partial J(\mathbf{w}_{t+1})$.

[1] Define $\bar{J}(\eta) := J(\mathbf{w}_i) - J(\mathbf{w}_i + \eta p)$ where $\mathbf{p} = -\frac{\nabla J(\mathbf{w}_i)}{\|\nabla J(\mathbf{w}_i)\|}$ is a unit-length gradient. We see that $\frac{d}{d\eta}\bar{J}(\eta)\Big|_{\eta=0} = [-\nabla J(\mathbf{w}_i + \eta\,\mathbf{p})^\top \mathbf{p}]|_{\eta=0} = \|\nabla J(\mathbf{w}_i)\|$, and $\bar{J}(0) = 0$. Hence Lemma B.3.2 is applicable in this case.

- For quadratic form regularization with PD matrix $\mathbf{M}$, *i.e.*, $\Omega^*(\boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\mu}\,\mathbf{M}^{-1}\,\boldsymbol{\mu}$, we have $r = \mathbf{g}_{t+1}^\top\,\mathbf{M}^{-1}\,\mathbf{g}_{t+1}$ where $\mathbf{g}_{t+1} \in \partial J(\mathbf{w}_{t+1})$.
- For unnormalized entropic regularization we have $\partial^2\Omega^*(\boldsymbol{\mu}) = \operatorname{diag}(e^{\mu_1}, \ldots, e^{\mu_d})$. Hence we may bound $r \leq \|\mathbf{g}_{t+1}\|_2^2 \exp(\|\boldsymbol{\mu}\|_\infty)$ where $\mathbf{g}_{t+1} \in \partial J(\mathbf{w}_{t+1})$. Clearly this bound may be very loose whenever $\boldsymbol{\mu}$ has only very few large coefficients.
- For normalized entropy regularization, *i.e.*, $\Omega^*(\boldsymbol{\mu}) = \log\sum_i \exp\mu_i$ we have $r \leq \|\mathbf{g}_{t+1}\|_\infty^2$ where $\mathbf{g}_{t+1} \in \partial J(\mathbf{w}_{t+1})$.

B.4 Proof of Theorem 3.2.2

Proof. For any $\epsilon_t > 4G^2/(\lambda\sigma)$ it follows from (3.6) that $\epsilon_{t+1} \leq \epsilon_t/2$. Hence we need at most $\log_2[\lambda\sigma\epsilon_1/(4G^2)]$ to achieve this level of precision. Subsequently we have

$$\epsilon_t - \epsilon_{t+1} \geq \frac{\lambda\sigma}{8G^2}\epsilon_t^2. \tag{B.13}$$

Invoking Lemma B.1.1 by setting $z = \lambda\sigma/(8G^2)$ and $p_1 = 4G^2/(\lambda\sigma)$ shows that $\epsilon_t \leq \epsilon$ after at most $8G^2/(\lambda\sigma\epsilon) - 1$ more iterations. This proves the first claim.

To analyze convergence in the second case we need to study two additional phases: for $\epsilon_t \in [H/2, 4G^2/(\lambda\sigma)]$ we see constant progress. Hence it takes us $4(\lambda\sigma)^{-2}[8G^2 - \lambda\sigma H]$ iterations to cover this interval. Finally in the third phase we have $\epsilon_{t+1} \leq \epsilon_t[1 - \lambda\sigma/(4H)]$. Starting from $\epsilon_t = H/2$ we need $\log_2[2\epsilon/H]/\log_2[1 - \lambda\sigma/(4H)]$ iterations to converge. Expanding the logarithm in the denominator close to 1 proves the claim. □

B.5 Proof of Theorem 3.2.3

We first note that the termination criterion of Algorithm 3 is slightly different from that of Algorithm 2. In order to apply the convergence results for Algorithm 2 to Algorithm 3 we redefine the following notations:

$$\begin{aligned} \epsilon_t &:= J(\mathbf{w}_{t+1}^b) - J_t(\mathbf{w}_{t+1}) \\ \mathbf{a}_{t+1} &\in \partial R(\mathbf{w}_{t+1}^c), \\ b_{t+1} &:= R(\mathbf{w}_{t+1}^c) - \langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1} \rangle\,, \end{aligned} \tag{B.14}$$

where

$$\begin{aligned} \mathbf{w}_{t+1}^b &:= \operatorname*{argmin}_\eta J(\mathbf{w}_t^b + \eta(\mathbf{w}_{t+1} - \mathbf{w}_t^b)), \quad \text{and} \\ \mathbf{w}_{t+1}^c &:= (1-\theta)\,\mathbf{w}_{t+1}^b + \theta\,\mathbf{w}_{t+1}\,. \end{aligned}$$

Albeit the difference in the iterate update rules of Algorithms 2 and 3, we show that the convergence proof for Algorithm 2 is applicable to Algorithm 3. We first provide some lemmas which help simplify the proof.

Firstly, the following lemma is crucial to the application of Lemma B.3.3 in the proof.

Lemma B.5.1. $J_{t+1}(\mathbf{w}_{t+1}) = \lambda\Omega(\mathbf{w}_{t+1}) + \langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1} \rangle + b_{t+1}$

Proof. $\mathbf{w}_{t+1}^b$ is the optimal value of J on the line joining $\mathbf{w}_{t+1}$ and $\mathbf{w}_t^b$ while $\mathbf{w}_{t+1}^c$ is a convex combination of $\mathbf{w}_{t+1}$ and $\mathbf{w}_{t+1}^b$. Moreover by definition of $\mathbf{a}_{t+1}$ and b_{t+1} we have $J(\mathbf{w}_{t+1}^c) = J_{t+1}(\mathbf{w}_{t+1}^c)$. Therefore,

$$J(\mathbf{w}_{t+1}^c) = J_{t+1}(\mathbf{w}_{t+1}^c) = \lambda\Omega(\mathbf{w}_{t+1}^c) + \left\langle \mathbf{a}_{t+1}, \mathbf{w}_{t+1}^c \right\rangle + b_{t+1} \geq J(\mathbf{w}_{t+1}^b). \tag{B.15}$$

But since Ω is convex

$$\Omega(\underbrace{(1-\theta)\,\mathbf{w}_{t+1}^b + \theta\,\mathbf{w}_{t+1}}_{\mathbf{w}_{t+1}^c}) \leq (1-\theta)\Omega(\mathbf{w}_{t+1}^b) + \theta\Omega(\mathbf{w}_{t+1}),$$

which can be rearranged to

$$\theta\Omega(\mathbf{w}_{t+1}^b) - \theta\Omega(\mathbf{w}_{t+1}) \leq \Omega(\mathbf{w}_{t+1}^b) - \Omega(\mathbf{w}_{t+1}^c).$$

Multiplying by λ and adding and subtracting $\theta R(\mathbf{w}_{t+1}^b)$ and $\theta\check{R}_t(\mathbf{w}_{t+1})$ to the above equation, we have,

$$\begin{aligned}
&\underbrace{\lambda\theta\Omega(\mathbf{w}_{t+1}^b) + \theta R(\mathbf{w}_{t+1}^b)}_{\theta J(\mathbf{w}_{t+1}^b)} - \underbrace{\lambda\theta\Omega(\mathbf{w}_{t+1}) - \theta\check{R}_t(\mathbf{w}_{t+1})}_{\theta J_t(\mathbf{w}_{t+1})} \\
&\leq \underbrace{\lambda\Omega(\mathbf{w}_{t+1}^b) + R(\mathbf{w}_{t+1}^b)}_{J(\mathbf{w}_{t+1}^b)} - \lambda\Omega(\mathbf{w}_{t+1}^c) - (1-\theta)R(\mathbf{w}_{t+1}^b) - \theta\check{R}_t(\mathbf{w}_t).
\end{aligned}$$

Plugging in (B.14) obtains

$$\theta\epsilon_t \leq J(\mathbf{w}_{t+1}^b) - \lambda\Omega(\mathbf{w}_{t+1}^c) - (1-\theta)R(\mathbf{w}_{t+1}^b) - \theta\check{R}_t(\mathbf{w}_{t+1}). \tag{B.16}$$

Putting (B.15) and (B.16) together

$$\left\langle \mathbf{a}_{t+1}, \mathbf{w}_{t+1}^c \right\rangle + b_{t+1} \geq J(\mathbf{w}_{t+1}^b) - \lambda\Omega(\mathbf{w}_{t+1}^c) \geq (1-\theta)R(\mathbf{w}_{t+1}^b) + \theta\check{R}_t(\mathbf{w}_{t+1}) + \theta\epsilon_t.$$

Since $\mathbf{w}_{t+1}^c = (1-\theta)\,\mathbf{w}_{t+1}^b + \theta\,\mathbf{w}_{t+1}$ it follows that

$$(1-\theta)\left\langle \mathbf{a}_{t+1}, \mathbf{w}_{t+1}^b \right\rangle + \theta\left\langle \mathbf{a}_{t+1}, \mathbf{w}_{t+1} \right\rangle + b_{t+1} \geq (1-\theta)R(\mathbf{w}_{t+1}^b) + \theta\check{R}_t(\mathbf{w}_{t+1}) + \theta\epsilon_t.$$

Which can be rearranged to

$$(1-\theta)\left(\left\langle \mathbf{a}_{t+1}, \mathbf{w}_{t+1}^{b}\right\rangle - R(\mathbf{w}_{t+1}^{b})\right) + \theta\left(\langle \mathbf{a}_{t+1}, \mathbf{w}_{t+1}\rangle - \check{R}_t(\mathbf{w}_{t+1})\right) + b_{t+1} \geq \theta\epsilon_t.$$

Since $\left\langle \mathbf{w}_{t+1}^{b}, \mathbf{a}_{t+1}\right\rangle + b_{t+1}$ is the Taylor approximation of the convex function R around $\mathbf{w}_{t+1}^{c}$ evaluated at $\mathbf{w}_{t+1}^{b}$ it follows that $R(\mathbf{w}_{t+1}^{b}) \geq \left\langle \mathbf{w}_{t+1}^{b}, \mathbf{a}_{t+1}\right\rangle + b_{t+1}$. Plugging this into the above equation yields

$$(1-\theta)(-b_{t+1}) + \theta(\langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1}\rangle - \check{R}_t(\mathbf{w}_{t+1})) + b_{t+1} \geq \theta\epsilon_t.$$

Dividing by $\theta > 0$ and rearranging yields

$$\langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1}\rangle + b_{t+1} \geq \check{R}_t(\mathbf{w}_t) + \epsilon_t.$$

The conclusion of the lemma follows from observing that

$$\begin{aligned}
\check{R}_{t+1}(\mathbf{w}_{t+1}) &= \max(\langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1}\rangle + b_{t+1}, \check{R}_t(\mathbf{w}_{t+1})) \\
&= \langle \mathbf{w}_{t+1}, \mathbf{a}_{t+1}\rangle + b_{t+1}, \text{ and that} \\
J_{t+1}(\mathbf{w}_{t+1}) &= \lambda\Omega(\mathbf{w}_{t+1}) + \check{R}_{t+1}(\mathbf{w}_{t+1})
\end{aligned}$$

□

We also need the following two lemmas before we can proceed to the final proof.

Lemma B.5.2. $\epsilon_t - \epsilon_{t+1} \geq J_{t+1}(\mathbf{w}_{t+2}) - J_t(\mathbf{w}_{t+1})$

Proof.

$$\begin{aligned}
\epsilon_t - \epsilon_{t+1} &= J(\mathbf{w}_{t+1}^{b}) - J_t(\mathbf{w}_{t+1}) - J(\mathbf{w}_{t+2}^{b}) + J_{t+1}(\mathbf{w}_{t+2}) \\
&= \underbrace{\left(J(\mathbf{w}_{t+1}^{b}) - J(\mathbf{w}_{t+2}^{b})\right)}_{\geq 0} + J_{t+1}(\mathbf{w}_{t+2}) - J_t(\mathbf{w}_{t+1}) \quad \text{(by the definition of } \mathbf{w}_i^b\text{)} \\
&\geq J_{t+1}(\mathbf{w}_{t+2}) - J_t(\mathbf{w}_{t+1})
\end{aligned}$$

□

Lemma B.5.3. *Let* $\boldsymbol{\alpha}_t$, $\mathbf{A}_{t+1} := [\mathbf{a}_1, \ldots, \mathbf{a}_{t+1}]$, *and* $\mathbf{b}_{t+1} := [b_1, \ldots, b_{t+1}]$ *be as defined in Lemma B.3.3. Then under the assumption of Theorem 3.2.1 that* $\|\mathbf{a}_i\| \leq G$, *we have*

$$[-\boldsymbol{\alpha}_t, 1]^\top \mathbf{A}_{t+1}^\top \mathbf{A}_{t+1}[-\boldsymbol{\alpha}_t, 1] \leq 4G^2.$$

Proof. By the dual connection, $\partial\lambda\Omega(\mathbf{w}_{t+1}) = -\mathbf{A}_t\,\boldsymbol{\alpha}_t$. Also, $\boldsymbol{\alpha}_t \geq \mathbf{0}$, and $\|\boldsymbol{\alpha}_t\|_1 = 1$ as it is the optimal solution of (3.4) at t-th iteration. It follows that $\partial\lambda\Omega(\mathbf{w}_{t+1})$ lies in the convex hull of $\mathbf{a}_i \in \partial R(\mathbf{w}_i^c)\ \forall i \leq t$. Therefore $\|\partial\lambda\Omega(\mathbf{w}_{t+1})\| \leq G$. Consequently,

by Cauchy-Schwarz inequality,

$$\begin{aligned}[-\boldsymbol{\alpha}_t, 1]^\top \mathbf{A}_{t+1}^\top \mathbf{A}_{t+1}[-\boldsymbol{\alpha}_t, 1] &= \|\partial\lambda\Omega(\mathbf{w}_{t+1}) + \mathbf{a}_{t+1}\|^2 \\ &= \|\partial\lambda\Omega(\mathbf{w}_{t+1})\|^2 + 2\partial\lambda\Omega(\mathbf{w}_{t+1})^\top \mathbf{a}_{t+1} + \|\mathbf{a}_{t+1}\|^2 \\ &\leq 4G^2.\end{aligned}$$

□

Finally, we sketch the proof for Theorem 3.2.3.

Proof. [Theorem 3.2.3] (Sketch) Theorem 3.2.1 holds for Algorithm 3 by applying Lemmas B.5.1, B.5.2, and B.5.3 into the first part of the proof. Therefore, for $\epsilon < 4G^2/(\lambda\sigma)$, (B.12) reduces to $\epsilon_t - \epsilon_{t+1} \geq \lambda\sigma\epsilon_t/(4G^2)$. Applying Lemma B.1.1 yields $\epsilon_t \leq \left(z\left(t - 1 + \frac{1}{\epsilon_1 z}\right)\right)^{-1}$, with $z = \lambda\sigma/(8G^2)$. Setting $\left(z\left(t - 1 + \frac{1}{\epsilon_1 z}\right)\right)^{-1} = \epsilon$, assuming that $\epsilon_1 > 0$, and solving for n yields $n \leq (z\epsilon)^{-1} = 8G^2/(\lambda\sigma\epsilon)$. □

Basic Definitions and Results in Convex Analysis

In this appendix we collect some basic definitions and results in convex analysis needed in this thesis. The materials presented in this section are due to Hiriart-Urruty and Lemaréchal [1993]. Similar materials can also be found in Rockafellar [1970].

Definition C.0.4 (Domain). *The domain of a function $J : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, not identically $+\infty$, denoted by* dom J, *is defined as*

$$\text{dom } J := \{\mathbf{w} \ : \ J(\mathbf{w}) < +\infty\} .$$

Definition C.0.5 (Convex set). *A set $C \subseteq \mathbb{R}^d$ is said to be convex if $\tau\, \mathbf{w}_1 + (1-\tau)\, \mathbf{w}_2$ is in C whenever $\mathbf{w}_1$ and $\mathbf{w}_2$ are in C, and $\tau \in [0,1]$.*

Definition C.0.6 (Convex function). *Let C be a nonempty convex set in $\mathbb{R}^d$. A function $J : C \to \mathbb{R}$ is said to be convex on C when, for all pairs $(\mathbf{w}_1, \mathbf{w}_2) \in C \times C$ and all $\tau \in (0,1)$, there holds*

$$J(\tau\, \mathbf{w}_1 + (1-\tau)\, \mathbf{w}_2) \leq \tau J(\mathbf{w}_1) + (1-\tau) J(\mathbf{w}_2) \ . \tag{C.1}$$

Furthermore, J is said to be strictly *convex on C when (C.1) holds as strict inequality if $\mathbf{w}_1 \neq \mathbf{w}_2$.*

Definition C.0.7 (Strongly convex function). *A function $J : C \to \mathbb{R}$ is σ-strongly convex on C with modulus $\sigma > 0$ if and only if the function $J - \frac{1}{2} \|\cdot\|$ is convex on C.*

Definition C.0.8 (Lipschitz continuous gradient). *A continuous function $J : \mathbb{R}^d \to \mathbb{R}$ is said to have L-Lipschitz continuous gradient with constant $L > 0$ if and only if*

$$\|\nabla J(\mathbf{w}_1) - \nabla J(\mathbf{w}_2)\| \leq L \|\mathbf{w}_1 - \mathbf{w}_2\| \quad \text{for all} \quad (\mathbf{w}_1, \mathbf{w}_2) \in \mathbb{R}^d \times \mathbb{R}^d .$$

Definition C.0.9 (Convex conjugate). *For a convex function $J : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, not identically $+\infty$, the conjugate function J^* is defined as*

$$\mathbb{R}^d \ni \boldsymbol{\mu} \mapsto J^*(\boldsymbol{\mu}) := \sup \{\langle \boldsymbol{\mu}, \mathbf{w} \rangle - J(\mathbf{w}) \ : \ \mathbf{w} \in \text{dom } J\} .$$

Theorem C.0.10 ([Hiriart-Urruty and Lemaréchal, 1993, Theorem X.4.1.1]). *Let $J : \mathbb{R}^d \to \mathbb{R}$ be strictly convex, then the conjugate J^* of J is continuously differentiable on the relative interior of* dom J.

Theorem C.0.11 ([Hiriart-Urruty and Lemaréchal, 1993, Theorem X.4.2.1]). *Assume $J : \mathbb{R}^d \to \mathbb{R}$ is strongly convex with modulus $\sigma > 0$ on $\mathbb{R}^d$: for all $(\mathbf{w}_1, \mathbf{w}_2) \in \mathbb{R}^d \times \mathbb{R}^d$ and $\tau \in (0, 1)$,*

$$J(\tau\,\mathbf{w}_1 + (1-\tau)\,\mathbf{w}_2) \leq \tau J(\mathbf{w}_1) + (1-\tau) J(\mathbf{w}_2) - \frac{\sigma}{2}\tau(1-\tau)\,\|\mathbf{w}_1 - \mathbf{w}_2\|^2 .$$

Then dom $J^* = \mathbb{R}^d$ *(where J^* is the conjugate of J) and ∇J^* is Lipschitzian with constant σ^{-1} on $\mathbb{R}^d$:*

$$\|\nabla J^*(\boldsymbol{\mu}_1) - \nabla J^*(\boldsymbol{\mu}_2)\| \leq \sigma^{-1}\,\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| \;\; \textit{for all } (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) \in \mathbb{R}^d \times \mathbb{R}^d .$$

Bibliography

N. Abe, J. Takeuchi, and M. K. Warmuth. Polynomial Learnability of Stochastic Rules with Respect to the KL-Divergence and Quadratic Distance. *IEICE Transactions on Information and Systems*, 84(3):299–316, 2001.

Y. Abu-Mostafa. A method for learning from hints. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 73–80, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*, volume 45. SIAM Classics in Applied Methematics, 2003.

G. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485, 1967.

G. Andrew and J. Gao. Scalable training of l_1-regularized log-linear models. In *Proceedings of the International Conference on Machine Learning*, pages 33–40, New York, NY, USA, 2007. ACM.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25:25–29, 2000.

G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. MIT Press, Cambridge, Massachusetts, 2007.

S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

P. L. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 113–120. MIT Press, Cambridge, MA, 2005.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the International Conference on Machine Learning*, pages 65–72, New York, NY, 2004. ACM Press.

A. Belloni. Introduction to bundle methods. Technical report, Operation Research Center, M.I.T., 2005.

A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods Software*, 1:23–34, 1992.

S. Benson, L. C. McInnes, J. Moré, T. Munson, and J. Sarich. TAO user manual (revision 1.9). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2007. http://www.mcs.anl.gov/tao.

D. P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21:174–184, 1976.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

C. Bhattacharyya, P. K. Shivaswamy, and A. J. Smola. A second order cone programming formulation for classifying missing data. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 153–160. MIT Press, Cambridge, MA, 2005.

J. Bi and T. Zhang. Support vector classification with input data uncertainty. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 161–168. MIT Press, Cambridge, MA, 2005.

S. Bickel. Discovery challenge 2006 overview. In *Proceedings of ECML/PKDD Discovery Challenge Workshop*, 2006.

A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *Proceedings of the International Conference on Machine Learning*, pages 89–96, New York, NY, USA, 2007. ACM.

A. Bordes, N. Usunier, and L. Bottou. Sequence labelling SVMs trained in one pass. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases: ECML PKDD 2008*, Lecture Notes in Computer Science, LNCS 5211, pages 146–161. Springer, 2008.

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proc. Annual Conf. Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Neural Information Processing Systems*. MIT Press, 2007.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures on Machine Learning (2004)*, pages 169–207, Berlin, 2004. Springer.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

J. S. Breese, D. Heckerman, and C. Kardie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

C. J. C. Burges. Geometry and invariance in kernel based methods. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 89–116, Cambridge, MA, 1999. MIT Press.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 78–87, New York, NY, USA, 2004. ACM Press.

J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. MIT Press, 2009.

O. Chapelle, Q. Le, and A. J. Smola. Large margin optimization of ranking measures. In *NIPS Workshop: Machine Learning for Web Search*, 2007.

Olivier Chapelle, Chuong B. Do, Quoc Le, Alex J. Smola, and Choon Hui Teo. Tighter bounds for structured estimation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 281–288. 2009.

E. W. Cheney and A. A. Goldstein. Newton's method for convex programming and Tchebycheff approximation. *Numerische Mathematik*, 1:253–268, 1959.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 158–169. Morgan Kaufmann, San Francisco, 2000.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *Journal of Machine Learning Research*, 9:1775–1822, 2008.

R. Collobert, F.H. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In W.W. Cohen and A. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006)*, pages 201–208. ACM, 2006.

A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. MPS-SIAM, 2000.

G. Cormack. TREC 2006 spam track overview. In *Fifteenth Text REtrieval Conference (TREC-2006)*, NIST, Gaithersburg, MD, 2006.

G. Cormack and T. Lynam. TREC 2005 spam track overview. In *Fourteenth Text REtrieval Conference (TREC-2005)*, NIST, Gaithersburg, MD, 2005.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

R. Cowell, A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Sytems*. Springer, New York, 1999.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, January 2003.

K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

N. A. C. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, New York, 1993.

D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002.

J. Duchi and Y. Singer. Online and batch learning using forward looking subgradients. In *Neural Information Processing Systems*, 2009. accepted.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *ICML*, 2008.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, New York, 2001. 2nd edition.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–451, 2004.

S. Elhedhli, J.-L. Goffin, and J.-P. Vial. Nondifferentiable optimization: Cutting plane methods. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2590–2595. Springer, 2008. 2nd edition.

R.-E. Fan, J.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, August 2008.

M. Ferraro and T. M. Caelli. Lie transformation groups, integral transforms, and invariant pattern recognition. *Spatial Vision*, 8:33–44, 1994.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In A. McCallum and S. Roweis, editors, *Proceedings of the International Conference on Machine Learning*, pages 320–327. Omnipress, 2008.

A. Frangioni. Generalized bundle methods. *SIAM Journal on Optimization*, 13(1): 117–156, 2002.

A. Frangioni. *Dual-Ascent Methods and Multicommodity Flow Problems*. PhD thesis, Dipartimento di Informatica, Universita' di Pisa, 1997. TD 5/97.

A. Globerson and S. Roweis. Nightmare at test time: Robust learning by feature deletion. In *International Conference on Machine Learning*, 2006.

A. Globerson, T. Koo, X. Carreras, and M. Collins. Exponentiated gradient algorithms for log-linear structured prediction. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 305–312, New York, NY, USA, 2007. ACM.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

T. Graepel and R. Herbrich. Invariant pattern recognition by semidefinite programming machines. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In *ICANN'98*, pages 201–206. Springer-Verlag, 1998.

W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2*. MIT Press, 1999.

N. Haarala, K. Miettinen, and M. Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109: 181–205, 2007.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *JMLR*, 5:1391–1415, 2004.

C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957.

G. E. Hinton. Learning translation invariant recognition in massively parallel networks. In *Proceedings Conference on Parallel Architectures and Laguages Europe*, pages 1–13. Springer, 1987.

J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms, I and II*, volume 305 and 306. Springer-Verlag, 1993.

J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen. Generalized hessian matrix and second-order optimality conditions for problems with $c^{1,1}$ data. *Applied Mathematics and Optimization*, 11:43–56, 1984.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear 2SVM. In W. Cohen, A. McCallum, and S. Roweis, editors, *Proceedings of the International Conference on Machine Learning*, pages 408–415. ACM, 2008.

A. N. Iusem and A. R. De Pierro. On the convergence properties of Hildreth's quadratic programming algorithm. *Mathematical Programming*, 47:37–51, 1990.

K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *ACM Special Interest Group in Information Retrieval (SIGIR)*, pages 41–48, New York, 2002. ACM.

T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, San Francisco, California, 2005. Morgan Kaufmann Publishers.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane traning of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. MIT Press, 2002. To Appear.

R. M. Karp. An algorithm to solve the $m \times n$ assignment problem in expected time $O(mn \log n)$. *Networks*, 10(2):143–152, 1980.

S. S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

J. E. Kelly. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, December 1960.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, University of California, Santa Cruz, Computer Research Laboratory, June 1994. Revised December 7, 1995. An extended abstract to appeared in the STOC 95, pp. 209–218.

K. C. Kiwiel. Efficiency of proximal bundle methods. *Journal of Optimization Theory and Applications*, 104:589–603, 2000.

K. C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.

K. C. Kiwiel. In *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Mathematics 1133, Berlin, 1985. Springer-Verlag.

K. C. Kiwiel. A dual method for certain positive semidefinite quadratic programming problems. *SIAM J. Sci. Stat. Comput.*, 10(1):175–186, 1989. ISSN 0196-5204.

K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46:105–122, 1990.

K. C. Kiwiel. A bundle Bregman proximal method for convex nondifferentiable minimization. *Mathematical Programming*, 85:241–258, 1999.

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *J. Mach. Learn. Res.*, 8:1519–1555, July 2007.

A. Kołcz and C. H. Teo. Feature weighting for improved classifier robustness. In *Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS'09)*, Mountain View, CA, USA, July 2009.

A. Kołcz and W.-T. Yih. Raising the baseline for high-precision text classifiers. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 400–409, New York, NY, USA, 2007. ACM.

I. R. Kondor. *Group Theoretical Methods in Machine Learning*. PhD thesis, Columbia University, New York City, NY, May 2008.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, volume 18, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In F. Fogelman-Soulié and P. Gallinari, editors, *Proceedings ICANN'95 — International Conference on Artificial Neural Networks*, volume II, pages 53–60, Nanterre, France, 1995. EC2.

C. Lemaréchal. Nonsmooth optimization and descent methods. Technical Report RR-78-004, IIASA, 2361 Laxenburg, Austria, 1978.

C. Lemaréchal. Numerical experiments in nonsmooth optimization. In E.A. Nurminski, editor, *Progress in Nondifferentiable Optimization*, pages 61–84. IIASA, Laxenburg, Austria, 1982.

C. Lemaréchal and C. Sagastizábal. Variable metric bundle methods: From conceptual to implementable forms. *Mathematical Programming*, 76:393–410, 1997.

C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–147, 1995.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, April 2008.

J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

M. M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17(1):1–29, 2002.

D. McAllester. Generalization bounds and consistency for structured labeling. In *Predicting Structured Data*, Cambridge, Massachusetts, 2007. MIT Press.

S. Mendelson. A few notes on statistical learning theory. In S. Mendelson and A. J. Smola, editors, *Advanced Lectures on Machine Learning*, number 2600 in LNAI, pages 1–40. Springer-Verlag, Heidelberg, Germany, 2003.

K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks ICANN'97*, volume 1327 of *Lecture Notes in Comput. Sci.*, pages 999–1004, Berlin, 1997. Springer-Verlag.

J. Munkres. Algorithms for the assignment and transportation problems. *Journal of SIAM*, 5(1):32–38, 1957.

A. Nedić. *Subgradient Methods for Convex Minimization*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2002.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

J. B. Orlin and Y. Lee. Quickmatch: A very fast algorithm for the assignment problem. Working Paper 3547-93, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, March 1993.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for $l_{1,\infty}$ regularization. In *Proceedings of the International Conference on Machine Learning*, 2009.

N. Ratliff, J. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proceedings of the International Conference on Machine Learning*, July 2006.

G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. J. Sommer, and B. Schölkopf. Improving the Caenorhabditis elegans genome annotation using machine learning. *PLoS Computational Biology*, 3(2):e20 doi:10.1371/journal.pcbi.0030020, 2007.

S. M. Robinson. Linear convergence of epsilon-subgradient descent methods for a class of convex functions. *Mathematical Programming*, 86:41–50, 1999.

R. T. Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series*. Princeton University Press, Princeton, NJ, 1970.

S. Rosset. Following curved regularized optimization solution paths. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, December 2004.

T. Sandler, J. Blitzer, P. P. Talukdar, and L. H. Ungar. Regularized learning with networks of features. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1401–1408, 2009.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks ICANN'96*, volume 1112 of *Lecture Notes in Computer Science*, pages 47–52, Berlin, 1996. Springer-Verlag.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal of Optimization*, 2:121–152, 1992.

S. Shalev-Schwartz and N. Srebro. SVM optimization: Inverse dependence on training set size. In W. Cohen, A. McCallum, and S. Roweis, editors, *Proceedings of the International Conference on Machine Learning*. Omnipress, 2008.

S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In H.U. Simon and G. Lugosi, editors, *Computational Learning Theory (COLT)*, LNCS. Springer, 2006. extended version.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the International Conference on Machine Learning*, 2007.

P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. A second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7, 2006.

N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, 1985.

P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufmann Publishers.

K. Sjöstrand and R. Larsen. The entire regularization path for the support vector domain description. In *Medical Image Computing and Computer-Assisted Intervention MICCAI 2006*, volume 4190 of *LCNS*, pages 241–248. Springer Berlin / Heidelberg, september 2006.

A. J. Smola, S. V. N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In R.G. Cowell and Z. Ghahramani, editors, *Proceedings of International Workshop on Artificial Intelligence and Statistics*, pages 325–332. Society for Artificial Intelligence and Statistics, 2005.

A. J. Smola, S. V. N. Vishwanathan, and Q. Le. Bundle methods for machine learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1377–1384. MIT Press, Cambridge, MA, 2008.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32, Cambridge, MA, 2004. MIT Press.

D. M. J. Tax and R. P. W. Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proceedings ESANN*, pages 251–256, Brussels, 1999. D Facto.

C. H. Teo, Q. V. Le, A. J. Smola, and S. V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736, New York, NY, USA, 2007. ACM.

C. H. Teo, A. Globerson, S. Roweis, and A. Smola. Convex learning with invariances. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1489–1496. MIT Press, Cambridge, MA, 2008.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Section B (Statistical Methodology)*, 58:267–288, 1996.

A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed problems*. V. H. Winston and Sons, 1977.

P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable optimization. *Mathematical Programming*, 117:387–423, 2009.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

H. Tuy. D.C. optimization: Theory, methods and algoritms. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, Dordrecht, The Netherlands, 1995. Kluwer Academic.

R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2008. 3rd edition.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16(2):264–281, 1971.

V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

V. Vapnik, S. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.

J. Vlček and L. Lukšan. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102(3):593–613, 1999.

U. von Luxburg and B. Schölkopf. Statistical learning theory: Models, concepts, and results. In D. M. Gabbay, S. Hartmann, and J. Woods, editors, *Handbook of the History of Logic: Inductive Logic*, volume 10. Elsevier North Holland, Amsterdam, Netherlands, 2009. In press.

E. Voorhees. Overview of the TREC 2001 question answering track. In *TREC*, 2001.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

G. Wang, D.-Y. Yeung, and F. H. Lochovsky. Two-dimensional solution path for support vector regression. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 993–1000, New York, NY, USA, 2006. ACM.

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599–621. Kluwer Academic, 1998.

C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *International Conference on Machine Learning (ICML)*, 2009.

J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization. In *Proceedings of the International Conference on Machine Learning*, 2008.

A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, 2003.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

J. Zhu, E. P. Xing, and B. Zhang. Primal sparse max-margin Markov networks. In *The 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD2009)*, 2009.